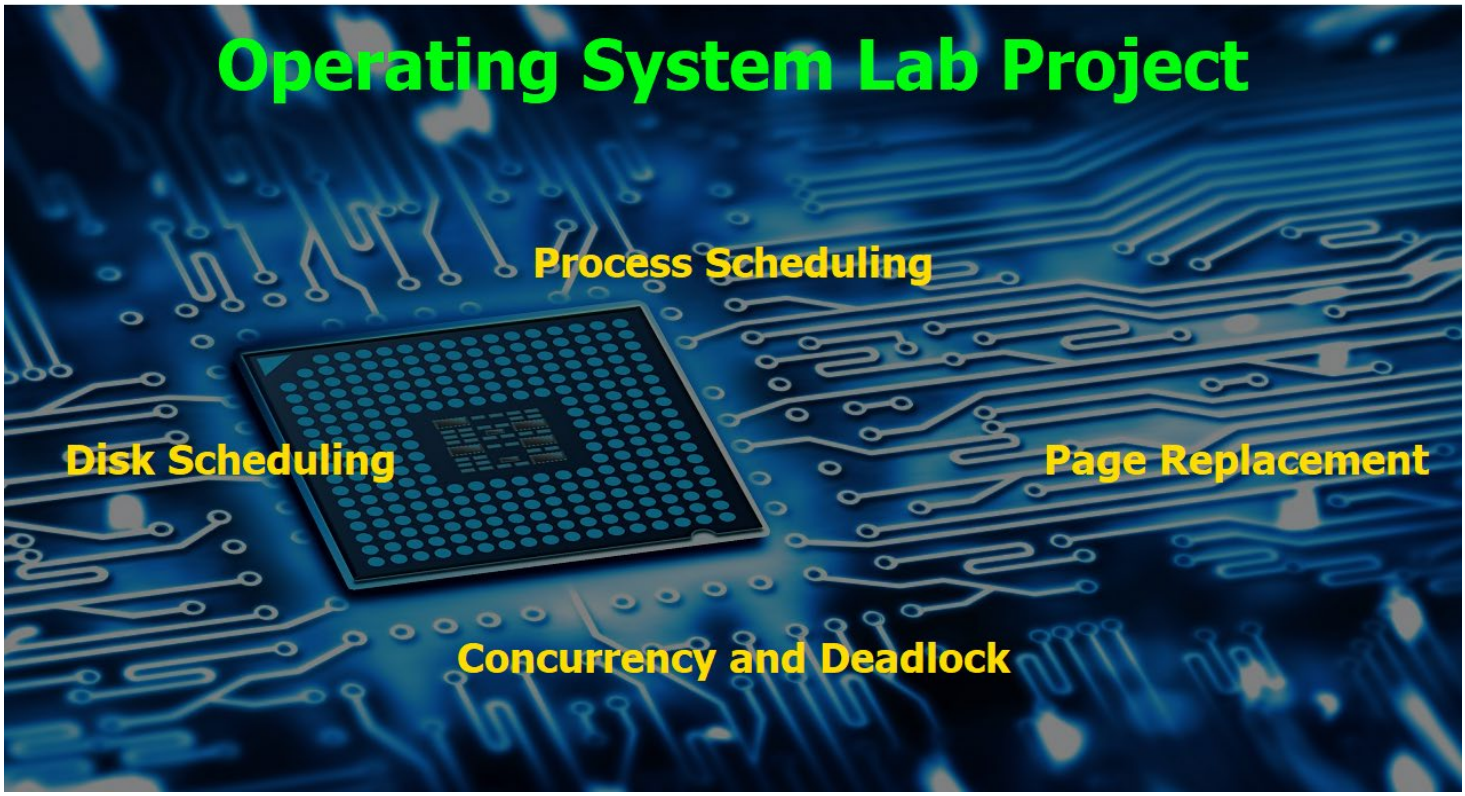
OS Virtual Lab



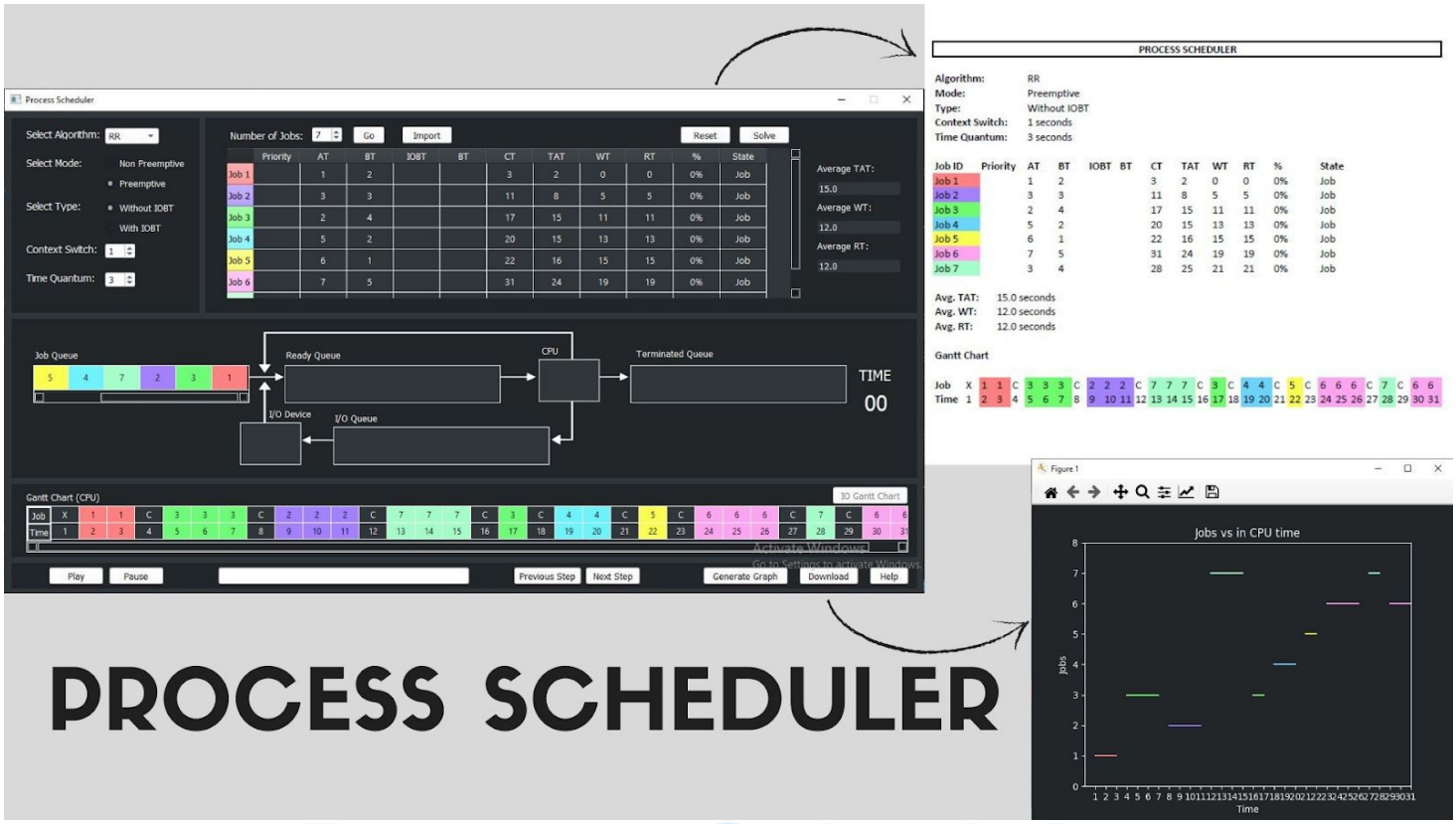
OS Virtual Lab

A Desktop Application to Visualize OS Lab Algorithms

Team - 13 & Team - 23

Developed By

Process Scheduler	Page Replacement	Disk Scheduler
Parth Prajapati	Rajan Gautam	Priyank Rana
Ritik Patel	Aman Mandal	Ratnam Shah
Nisarg Kapkar	Harshit Rathi	Ravi Trivedi
Purv Patel	Oday Alashoush	Rushil Vora
	Ugra Rajbanshi	Vismay Gajera



Process Scheduler

First Come First Serve (FCFS)

Introduction

Simplest scheduling algorithm that schedules according to arrival times of processes. FCFS scheduling algorithm states that the process that requests the CPU first is allocated the CPU first. It is implemented by using the FIFO queue. When a process enters the ready queue, its PCB is linked to the tail of the queue. When the CPU is free, it is allocated to the process at the head of thequeue. The running process is then removed from the queue. FCFS is a non-preemptive scheduling algorithm.

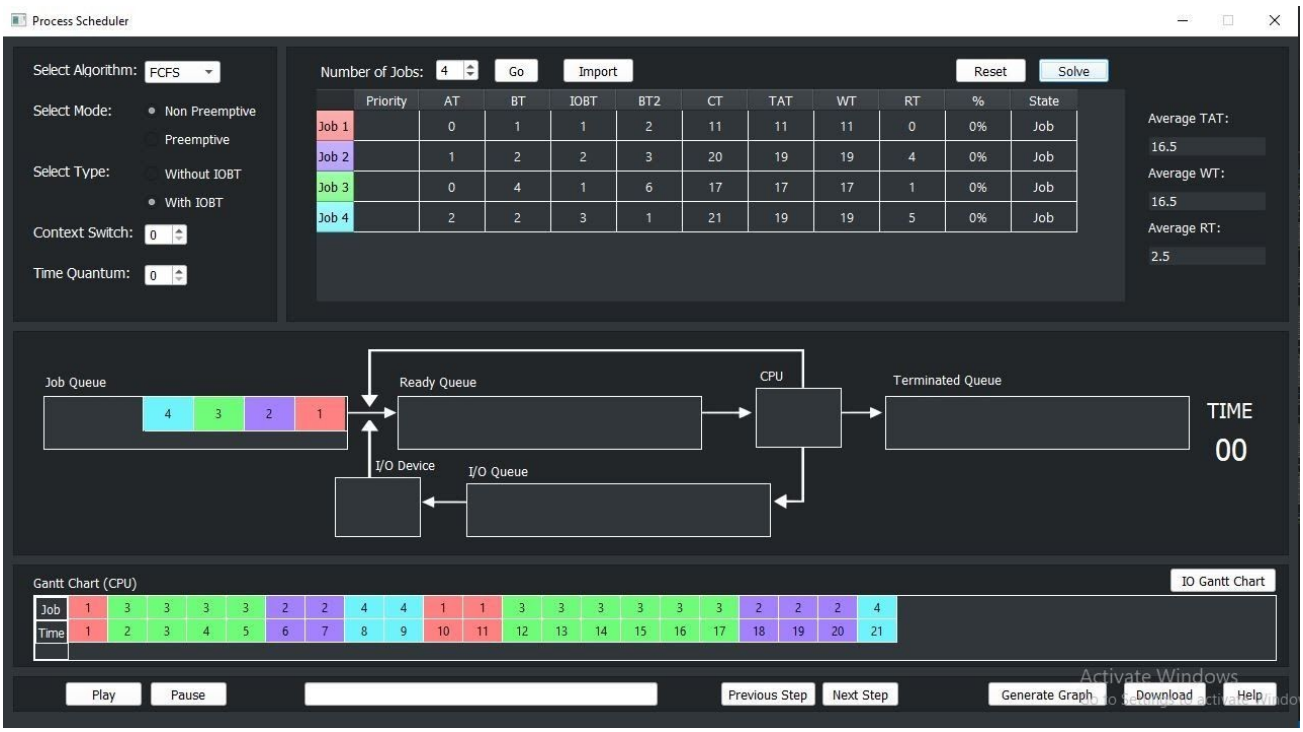
Advantages

- It is simple and easy to understand.

Disadvantages

- The process with less execution time suffers i.e., waiting time is often quite long and favors CPU Bound process then I/O bound process.
- Now, suppose the first process has a large burst time, and other processes have less burst time, then the processes will have to wait more unnecessarily, this will result in more average waiting time (Convoy Effect), which results in lower CPU and device utilization.

Snapshots



Shortest Job First (SJF)

Introduction

It is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be

preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution.

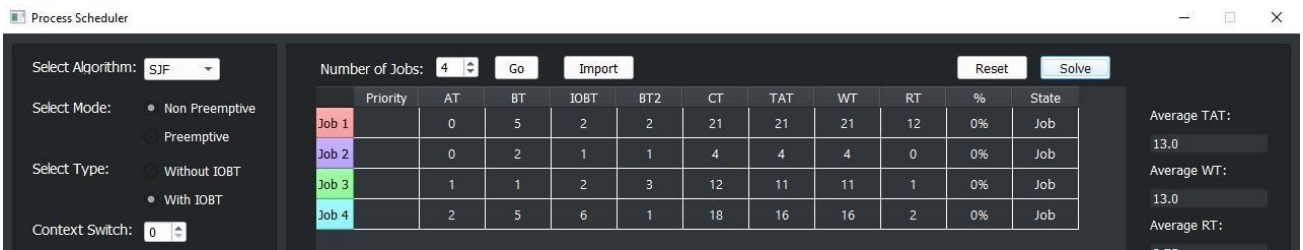
Advantages

- Shortest jobs are favored.
- It is probably optimal; in that it gives the minimum average waiting time for a given set of processes.

Disadvantages

- SJF may cause starvation if shorter processes keep coming. This problem is solved by aging.
- It cannot be implemented at the level of short-term CPU scheduling.

Snapshots



Shortest Remaining Time First (SRTF)

Introduction

This Algorithm is the preemptive version of SJF scheduling. In SRTF, the execution of the process can be stopped after a certain amount of time. At the arrival of every process, the short-term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process. Once all the processes are available in the ready queue.

Advantages

- Shortest jobs are favored.
- It is probably optimal; in that it gives the minimum average waiting time for a given set of processes.

Disadvantages

- SRTF may cause starvation if shorter processes keep coming. This problem is solved by aging.
- It cannot be implemented at the level of short-term CPU scheduling.

Snapshots

Round Robin (RR)

Introduction

It is the preemptive process scheduling algorithm. Each process is provided a fixed time to execute, it is called a quantum.

Once a process is executed for a given time period, it is preempted and another process executes for a given time period. Context switching is used to save states of preempted processes.

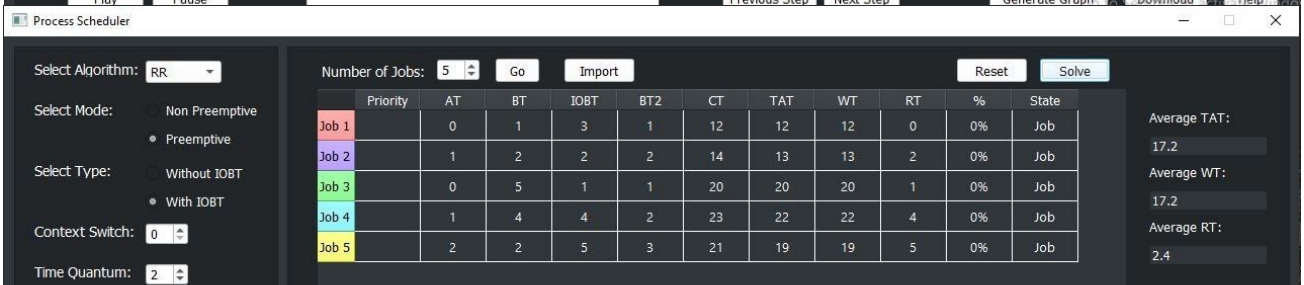
Advantages

- Every process gets an equal share of the CPU.
- RR is cyclic in nature, so there is no starvation.

Disadvantages

- Setting the quantum too short increases the overhead and lowers the CPU efficiency, but setting it too long leads to poor responses to short processes.
- The average waiting time under the RR policy is often long.

Snapshots



Longest Job First (LJF)

Introduction

It is similar to the SJF scheduling algorithm. But, in this scheduling algorithm, we givepriority to the process having the longest burst time. This is non-preemptive in nature i.e., when any process starts executing, can't beinterrupted before complete execution.

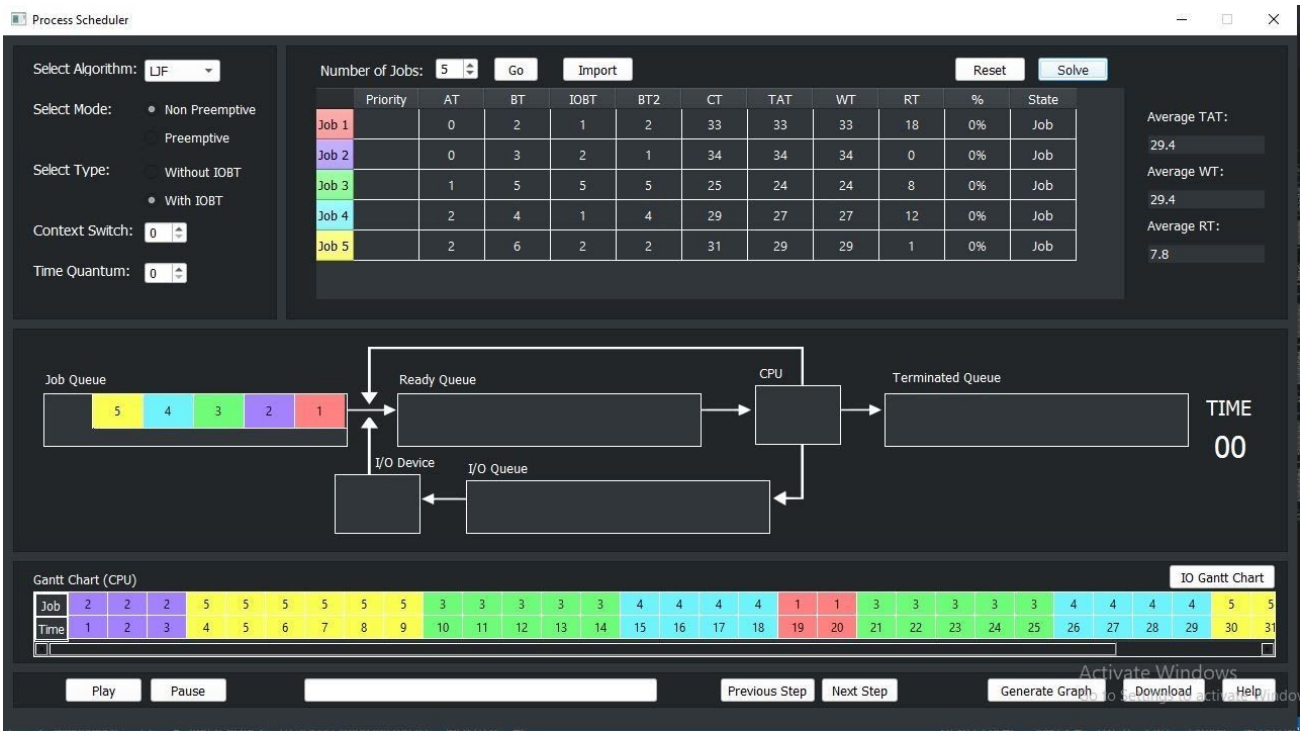
Advantages

- No process can complete until the longest job also reaches its completion.
- All the processes approximately finish at the same time.

Disadvantages

- The waiting time is high and processes with smaller burst time may starve for the CPU.

Snapshots



Longest Remaining Time First (LRTF)

Introduction

This Algorithm is the preemptive version of LJF scheduling. In this scheduling algorithm, we find the process with the maximum remaining time and then process it. We check for the maximum remaining time after some interval of time (say 1 unit each) to check if another process having more Burst Time arrived up to that time.

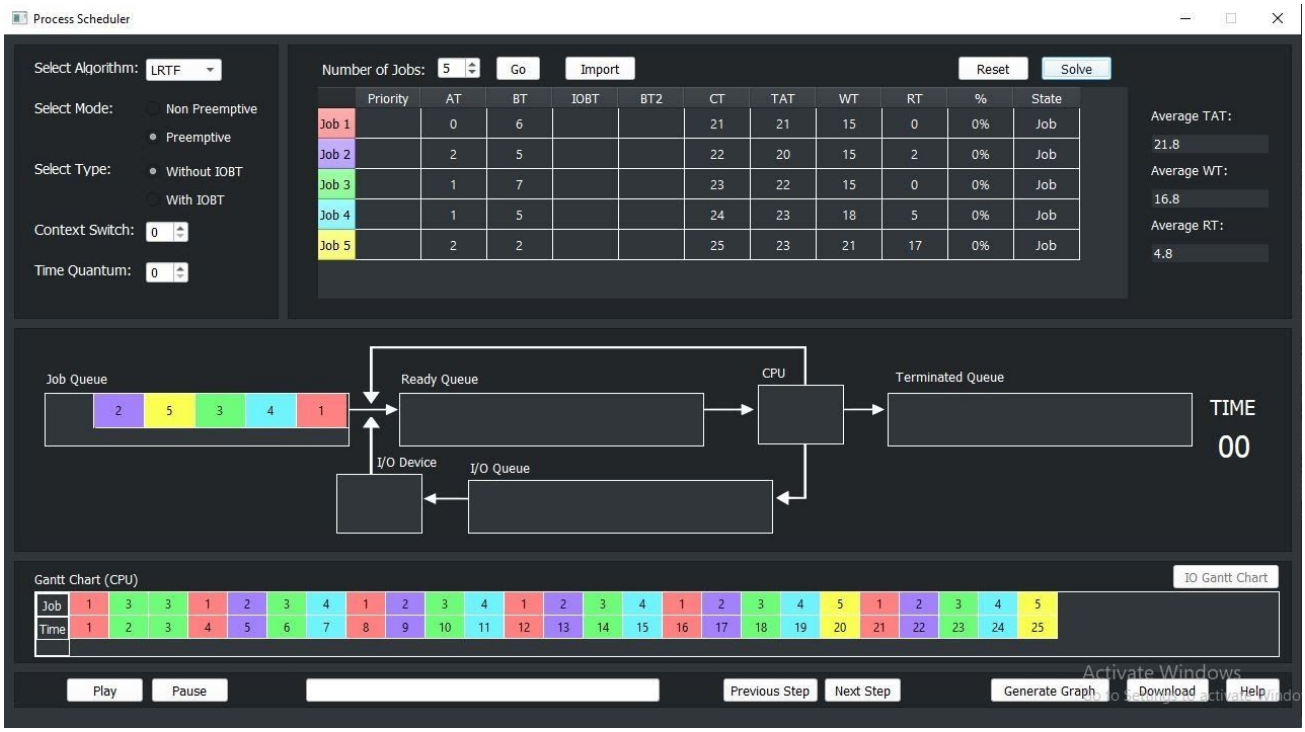
Advantages

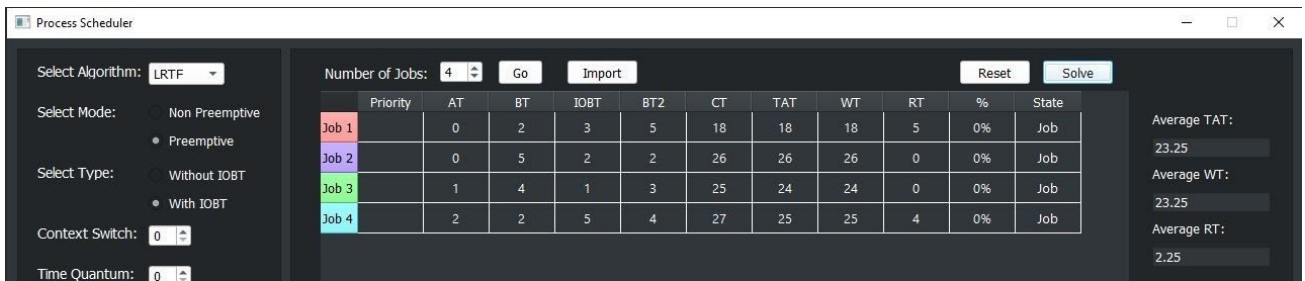
- No process can complete until the longest job also reaches its completion.
- All the processes approximately finish at the same time.

Disadvantages

- The waiting time is high and processes with smaller burst time may starve for the CPU.

Snapshots





Priority Based Scheduling

Introduction

In this scheduling, processes are scheduled according to their priorities, i.e., the highest priority process is scheduled first. If the priorities of two processes match, then schedule according to arrival time. Here starvation of process is possible.

Advantages

- This provides a good mechanism where the relative importance of each process may be precisely defined.

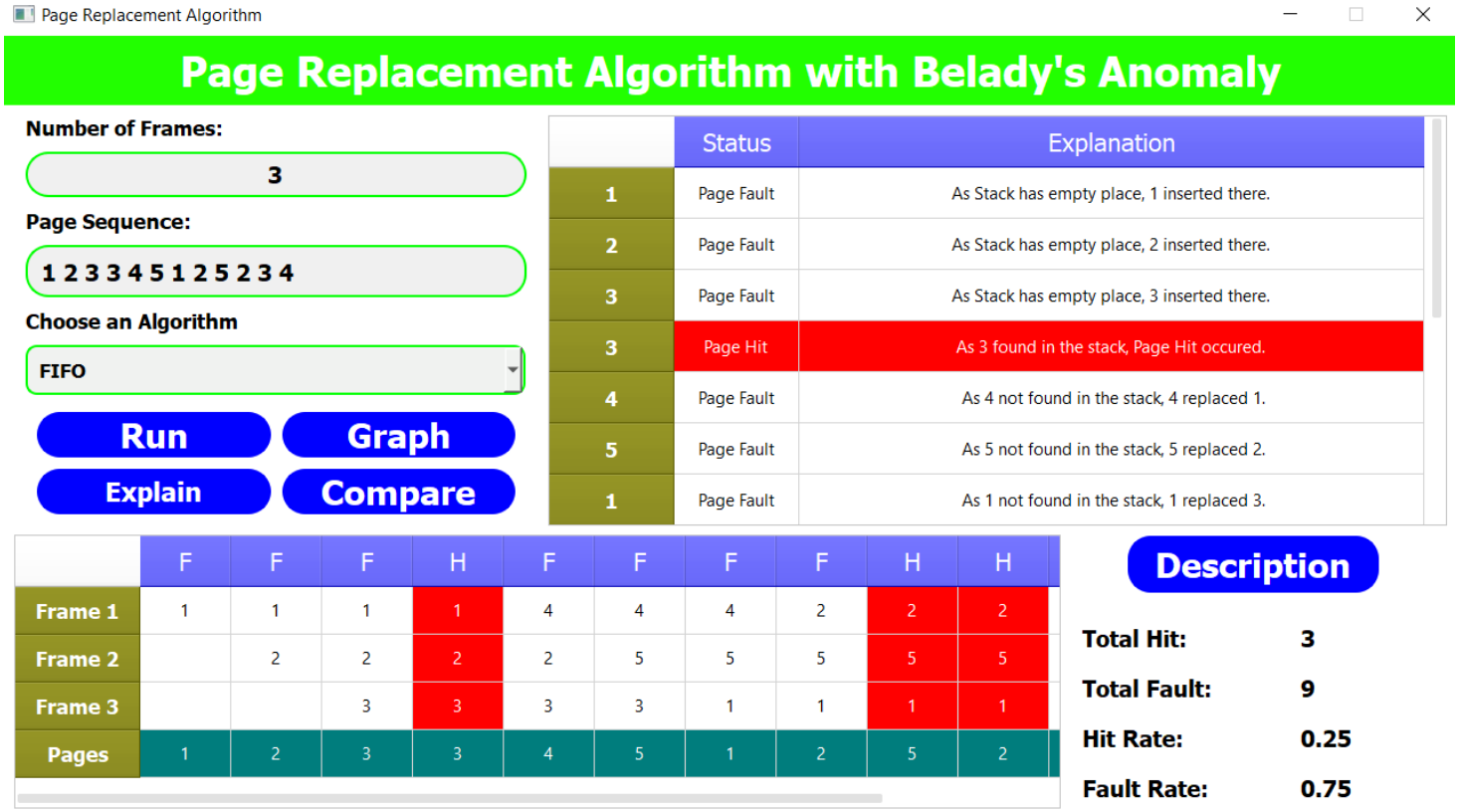
Disadvantages

- If high priority processes use up a lot of CPU time, lower priority processes may starve and be postponed indefinitely. The situation where a process never gets scheduled to run is called Starvation.
- Another problem is deciding which process gets which priority level assigned to it.

Snapshots


Process Scheduler
Select Algorithm: Priority
Select Mode: Non Preemptive
Preemptive
Select Type: Without IOBT
With IOBT
Context Switch: 1
Time Quantum: 0
Number of Jobs: 5
Go
Import
Reset
Solve
Priority AT BT IOBT BT2 CT TAT WT RT % State
Job 1 2 0 2 2 2 0 0 0% Job
Job 2 3 0 5 18 18 13 13 0% Job
Job 3 0 2 6 9 7 1 1 0% Job
Job 4 1 1 2 12 11 9 9 0% Job
Job 5 4 1 4 23 22 18 18 0% Job
Average TAT: 12.0
Average WT: 8.2
Average RT: 8.2
Job Queue
3 5 4 2 1
Ready Queue
CPU
Terminated Queue
TIME
00
I/O Device
I/O Queue
Gantt Chart (CPU)
IO Gantt Chart
Job 1 1 C 3 3 3 3 3 3 C 4 4 C 2 2 2 2 2 C 5 5 5 5
Time 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Play
Pause
Previous Step
Next Step
Generate Graph
Download
Help
Activate Windows


Process Scheduler
Select Algorithm: Priority
Select Mode: Non Preemptive
Preemptive
Select Type: Without IOBT
With IOBT
Context Switch: 0
Time Quantum: 0
Number of Jobs: 4
Go
Import
Reset
Solve
Priority AT BT IOBT BT2 CT TAT WT RT % State
Job 1 1 0 5 2 4 14 14 14 2 0% Job
Job 2 0 0 2 1 1 4 4 4 0 0% Job
Job 3 3 2 3 3 2 29 27 27 14 0% Job
Job 4 2 1 4 2 5 23 22 22 7 0% Job
Average TAT: 16.75
Average WT: 16.75
Average RT: 5.75
Job Queue
4 3 2 1
Ready Queue
CPU
Terminated Queue
TIME
00
I/O Device
I/O Queue
Gantt Chart (CPU)
IO Gantt Chart
Job 2 2 1 2 1 1 1 1 4 4 1 1 1 1 4 4 3 3 4 4 4 4 4 3 X X X 3 3
Time 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Play
Pause
Previous Step
Next Step
Generate Graph
Download
Help
Activate Windows



Page Replacement

In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

Paging - Paging is a process of reading data from, and writing data to, the secondary storage. It is a memory management scheme that is used to retrieve processes from the secondary memory in the form of pages and store them in the primary memory. The main objective of paging is to divide each process in the form of pages of fixed size. These pages are stored in the main memory in frames. Pages of a process are only brought from the secondary memory to the main memory when they are needed.

When an executing process refers to a page, it is first searched in the main memory. If it is not present in the main memory, a page fault occurs.

Page Fault – A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.
Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Page Replacement Algorithms

Page Replacement Algorithm decides which page to remove, also called swap out when a new page needs to be loaded into the main memory. Page Replacement happens when a requested page is not present in the main memory and the available space is not sufficient for allocation to the requested page.

When the page that was selected for replacement was paged out, and referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

A page replacement algorithm tries to select which pages should be replaced so as to minimize the total number of page misses. There are many different page replacement algorithms. These algorithms are evaluated by running them on a particular string of memory reference and computing the number of page faults. The fewer is the page faults the better is the algorithm for that situation.

First in First Out (FIFO)

Introduction

This is the simplest page replacement algorithm. In this algorithm, the OS maintains a queue that keeps track of all the pages in memory, with the oldest page at the front and the most recent page at the back.

When there is a need for page replacement, the FIFO algorithm, swaps out the page at the front of the queue, that is the page which has been in the memory for the longest time.

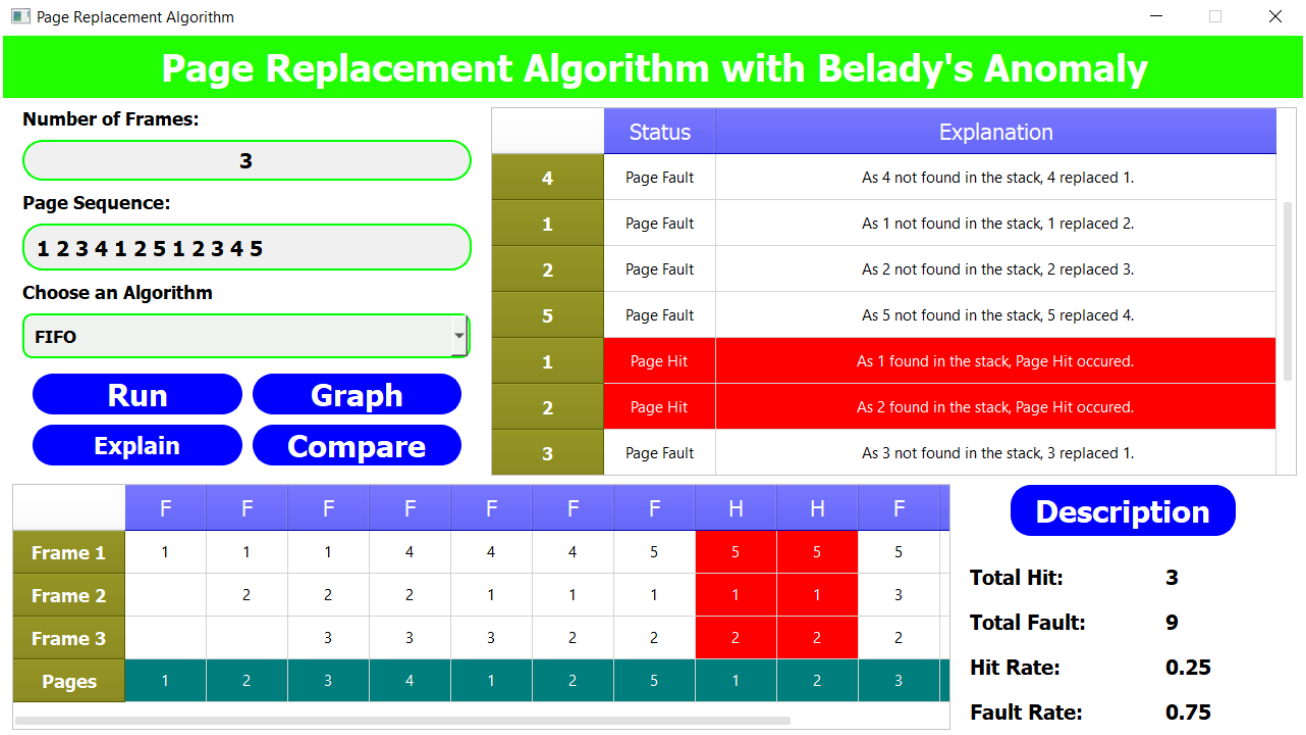
Advantages

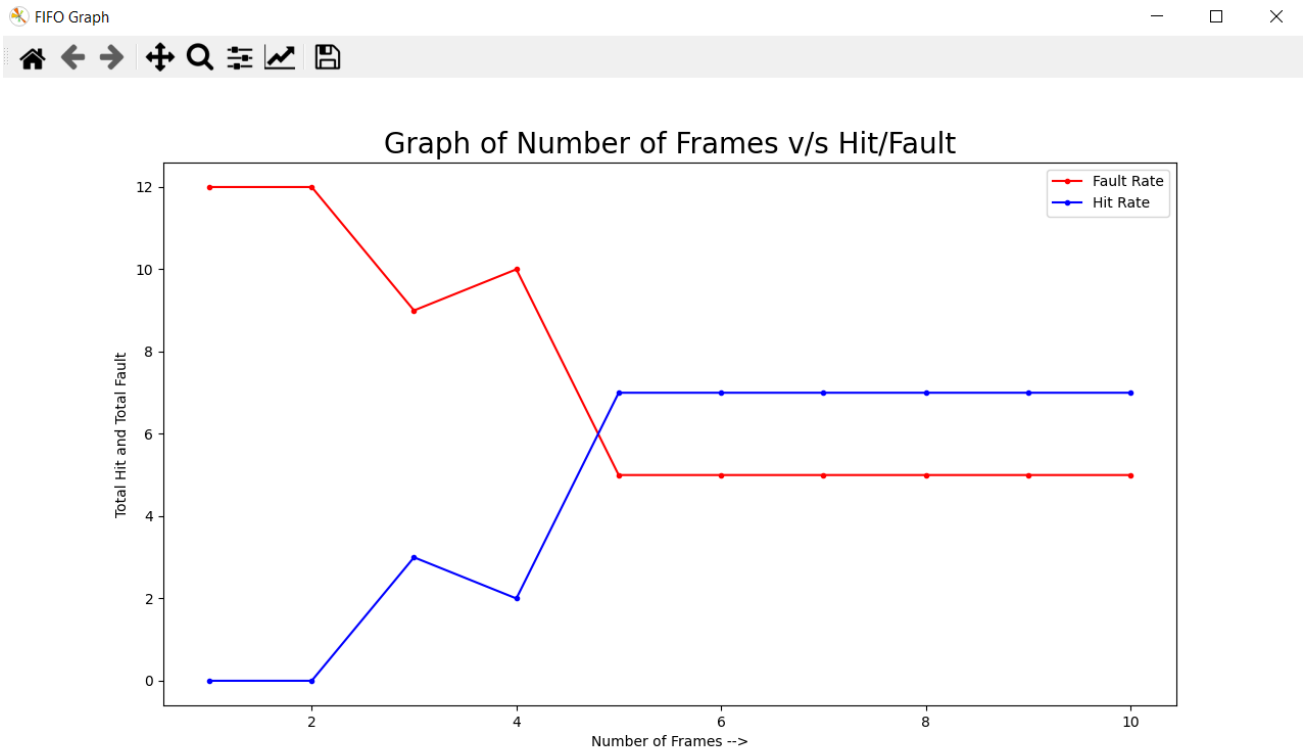
- Simple and easy to implement.
- Low overhead.

Disadvantages

- Poor performance.
- Doesn't consider the frequency of use or last used time, simple replaces the oldest page.
- Suffers from Belady's Anomaly (i.e., more page faults when we increase the number of page frames).

Snapshots


Page Replacement Algorithm
Page Replacement Algorithm with Belady's Anomaly
Number of Frames:
3
Page Sequence:
1 2 3 4 1 2 5 1 2 3 4 5
Choose an Algorithm
FIFO
Run
Graph
Explain
Compare
Status
Explanation
4 Page Fault As 4 not found in the stack, 4 replaced 1.
1 Page Fault As 1 not found in the stack, 1 replaced 2.
2 Page Fault As 2 not found in the stack, 2 replaced 3.
5 Page Fault As 5 not found in the stack, 5 replaced 4.
1 Page Hit As 1 found in the stack, Page Hit occured.
2 Page Hit As 2 found in the stack, Page Hit occured.
3 Page Fault As 3 not found in the stack, 3 replaced 1.
F F F F F F F H H F
Frame 1 1 1 1 4 4 4 5 5 5 5
Frame 2 2 2 2 1 1 1 1 1 3
Frame 3 3 3 3 2 2 2 2 2
Pages 1 2 3 4 1 2 5 1 2 3
Description
Total Hit: 3
Total Fault: 9
Hit Rate: 0.25
Fault Rate: 0.75


FIFO Graph
Graph of Number of Frames v/s Hit/Fault
Fault Rate
Hit Rate
Total Hit and Total Fault
Number of Frames -->

Page Replacement Algorithm

Page Replacement Algorithm with Belady's Anomaly

Number of Frames:

4

Page Sequence:

1 2 8 9 2 3 4 1 2 4 5 1 2 7 3 4 5

Choose an Algorithm

FIFO

Run | Graph | Explain | Compare

	Status	Explanation
1	Page Fault	As Stack has empty place, 1 inserted there.
2	Page Fault	As Stack has empty place, 2 inserted there.
8	Page Fault	As Stack has empty place, 8 inserted there.
9	Page Fault	As Stack has empty place, 9 inserted there.
2	Page Hit	As 2 found in the stack, Page Hit occured.
3	Page Fault	As 3 not found in the stack, 3 replaced 1.
4	Page Fault	As 4 not found in the stack, 4 replaced 2.

	F	F	F	F	H	F	F	F	F	H
Frame 1	1	1	1	1	1	3	3	3	3	3
Frame 2		2	2	2	2	2	4	4	4	4
Frame 3			8	8	8	8	8	1	1	1
Frame 4				9	9	9	9	9	2	2

Description

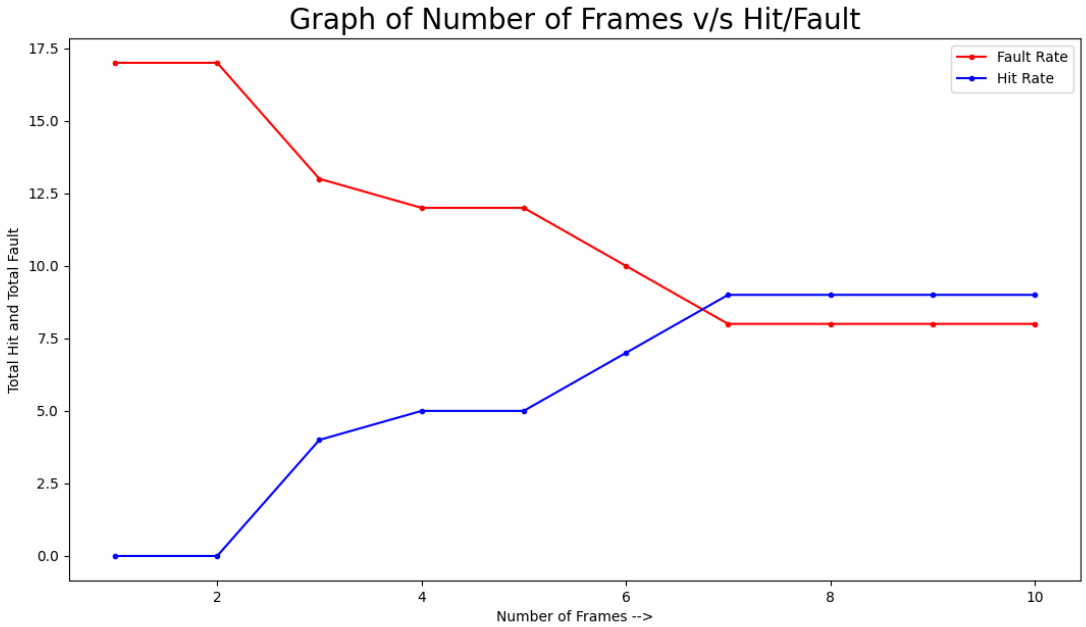
Total Hit: **5**

Total Fault: **12**

Hit Rate: **0.29**

Fault Rate: **0.71**

FIFO Graph



Last In First Out (LIFO)

Introduction

This Page Replacement algorithm stands for "Last In First Out". This algorithm works in a similar way to the LIFO principle. In this, the newest page is replaced which is arrived at last in the primary memory. This algorithm makes use of the stack for monitoring all the pages. It is implemented by keeping track of all the pages in a stack.

Advantages

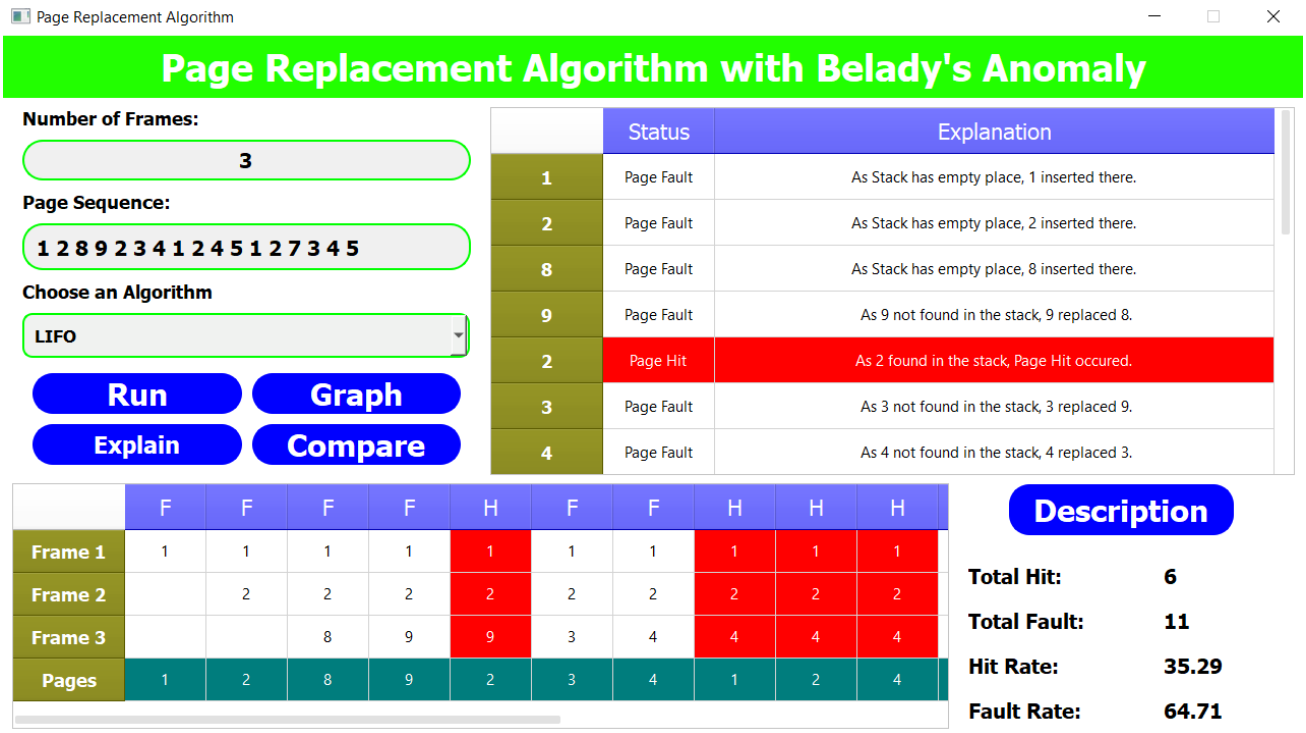
- Simple and easy to implement.
- Low overhead.

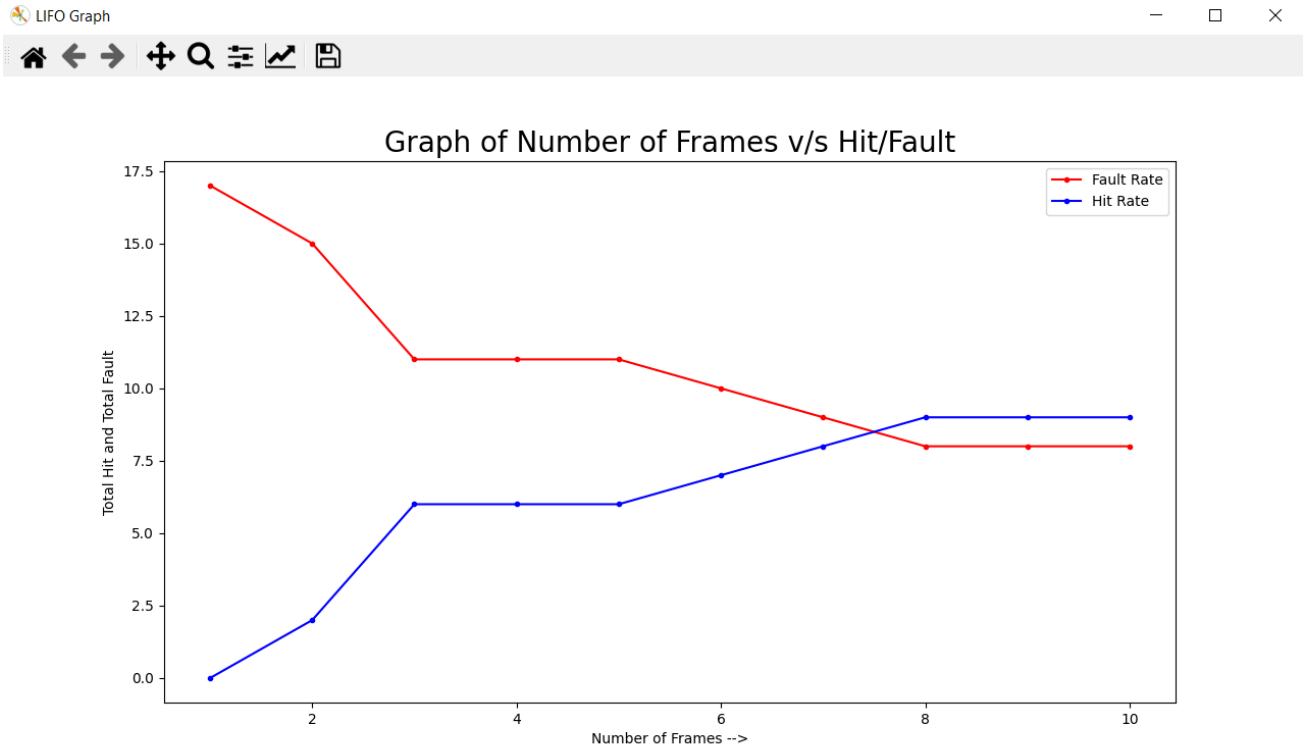
Disadvantages

- Poor performance.
- Doesn't consider the frequency of use or last used time, simple replaces the newest page.

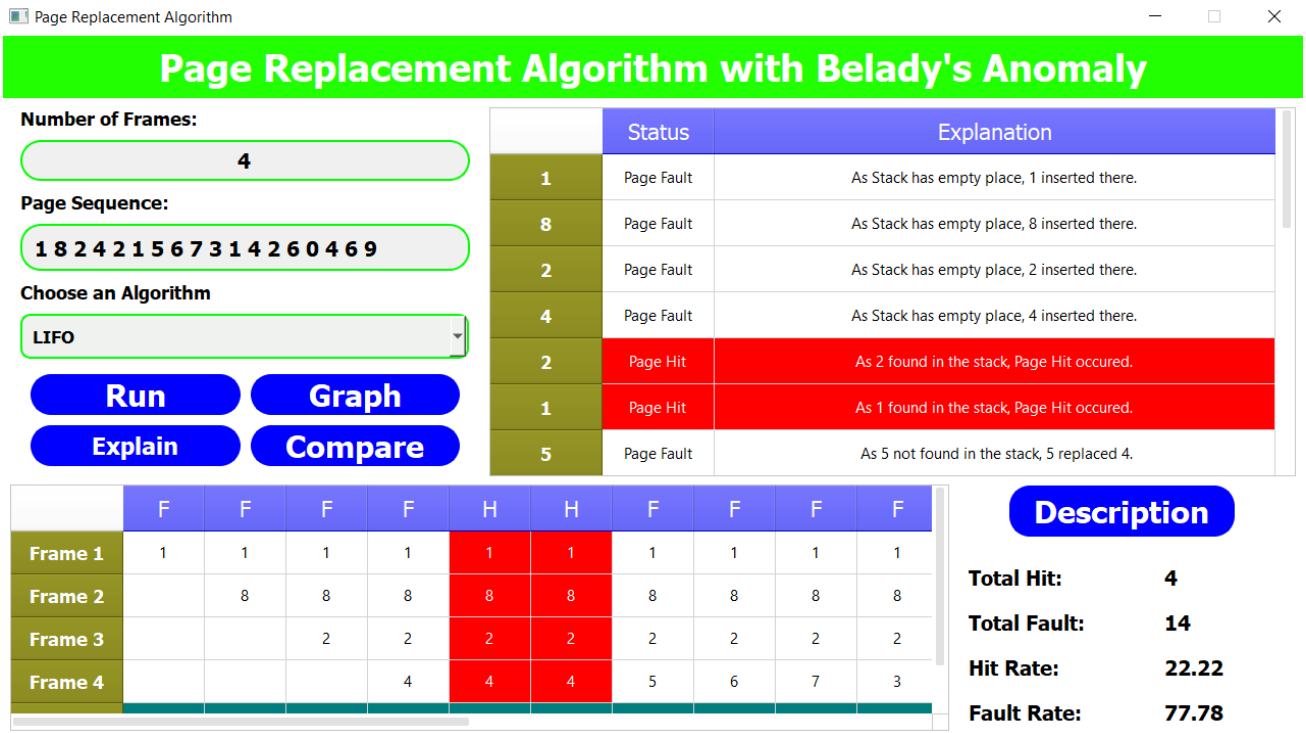
Snapshots

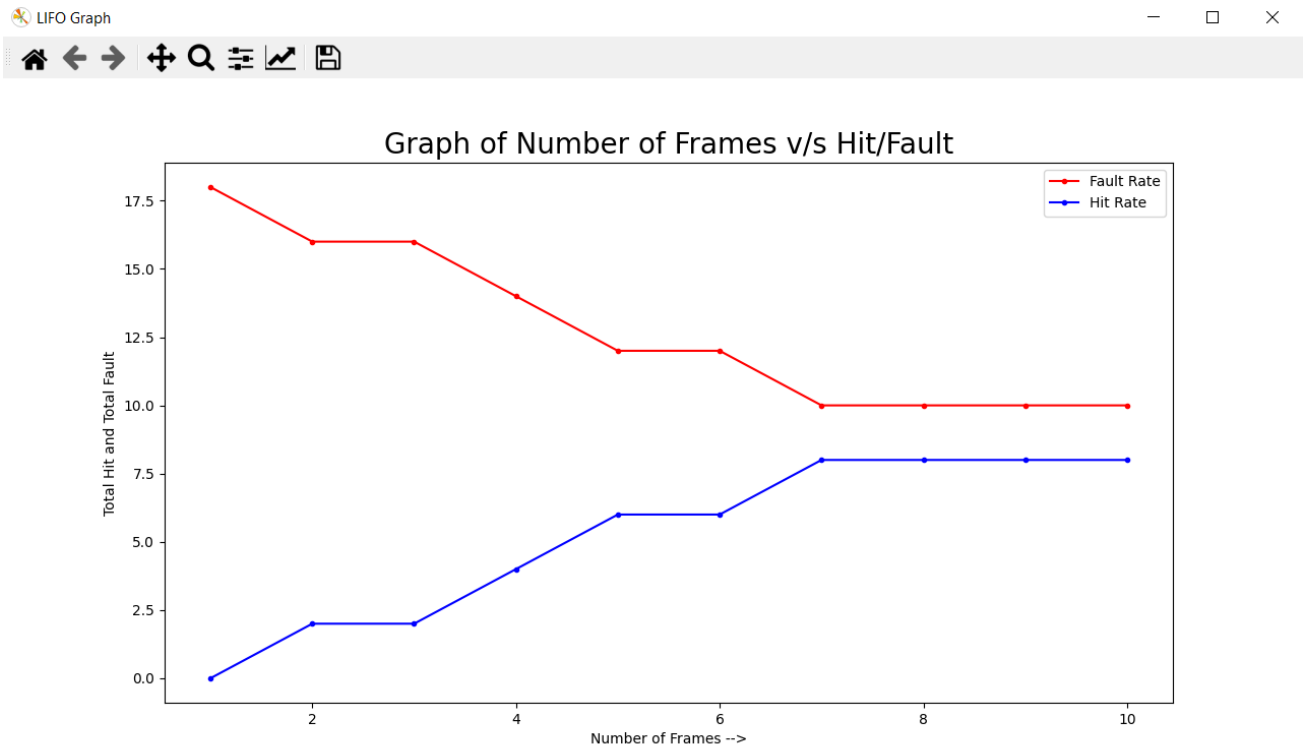
Example 1




LIFO Graph
Graph of Number of Frames v/s Hit/Fault
Fault Rate
Hit Rate
Total Hit and Total Fault
Number of Frames -->

Example 2


Page Replacement Algorithm
Page Replacement Algorithm with Belady's Anomaly
Number of Frames:
4
Page Sequence:
1 8 2 4 2 1 5 6 7 3 1 4 2 6 0 4 6 9
Choose an Algorithm
LIFO
Run
Graph
Explain
Compare
Status
Explanation
1 Page Fault As Stack has empty place, 1 inserted there.
8 Page Fault As Stack has empty place, 8 inserted there.
2 Page Fault As Stack has empty place, 2 inserted there.
4 Page Fault As Stack has empty place, 4 inserted there.
2 Page Hit As 2 found in the stack, Page Hit occured.
1 Page Hit As 1 found in the stack, Page Hit occured.
5 Page Fault As 5 not found in the stack, 5 replaced 4.
F F F F H H F F F F
Frame 1 1 1 1 1 1 1 1 1 1 1
Frame 2 8 8 8 8 8 8 8 8 8
Frame 3 2 2 2 2 2 2 2 2
Frame 4 4 4 4 5 6 7 3
Description
Total Hit: 4
Total Fault: 14
Hit Rate: 22.22
Fault Rate: 77.78



Least Recently Used (LRU)

Introduction

Least Recently Used page replacement algorithm keeps track of page usage over a short period of time. It works on the idea that the pages that have been most heavily used in the past are most likely to be used heavily in the future too.

In LRU, whenever page replacement happens, the page which has not been used for the longest amount of time is replaced.

Advantages

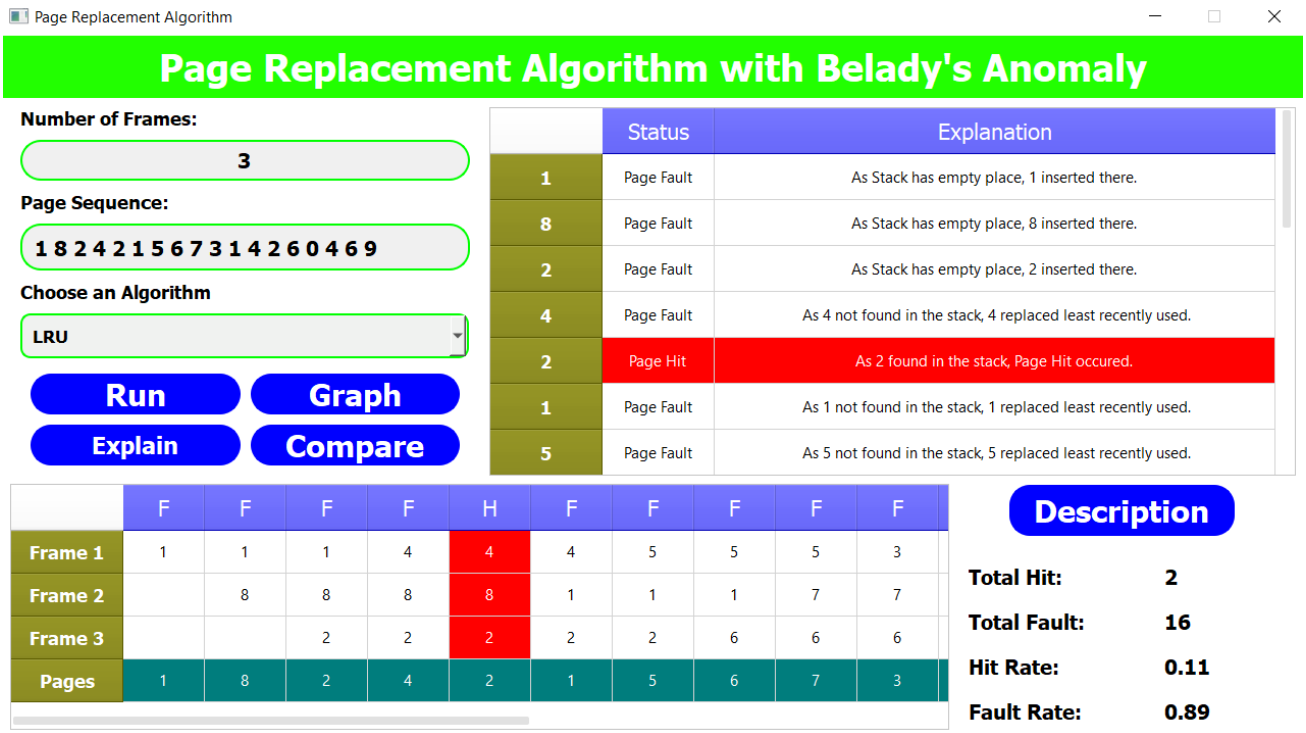
- Efficient.
- Don’t suffer from Belady’s Anomaly.

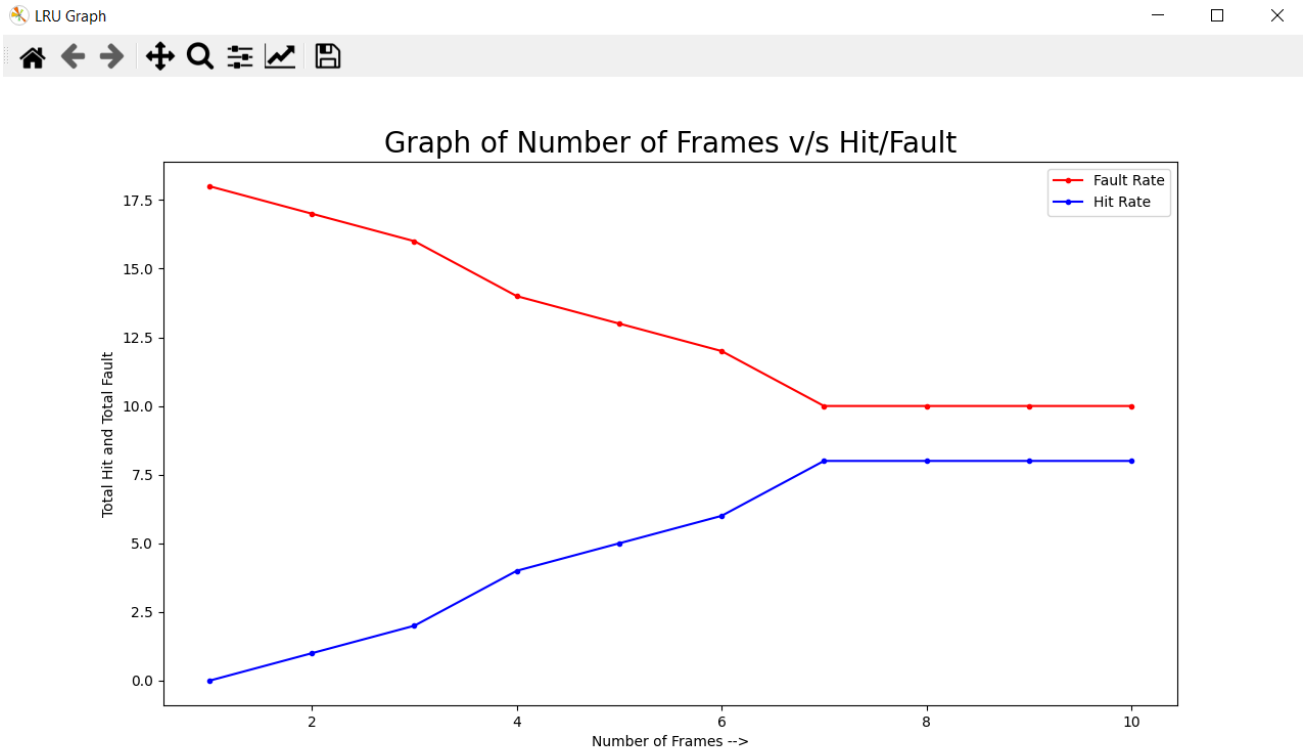
Disadvantages

- Complex implementation.
- Expensive.
- Requires hardware support.

Snapshots

Example 1





Example 2

Page Replacement Algorithm

Page Replacement Algorithm with Belady's Anomaly

Number of Frames:

4

Page Sequence:

4 5 6 7 4 1 2 0 3 1 4 5 2 4 7 8 9 6 4 1 0 2

Choose an Algorithm

LRU

Run | Graph | Explain | Compare

	Status	Explanation
4	Page Fault	As Stack has empty place, 4 inserted there.
5	Page Fault	As Stack has empty place, 5 inserted there.
6	Page Fault	As Stack has empty place, 6 inserted there.
7	Page Fault	As Stack has empty place, 7 inserted there.
4	Page Hit	As 4 found in the stack, Page Hit occured.
1	Page Fault	As 1 not found in the stack, 1 replaced least recently used.
2	Page Fault	As 2 not found in the stack, 2 replaced least recently used.

	F	F	F	F	H	F	F	F	F	H
Frame 1	4	4	4	4	4	4	4	4	3	3
Frame 2		5	5	5	5	1	1	1	1	1
Frame 3			6	6	6	6	2	2	2	2
Frame 4				7	7	7	7	0	0	0

Description

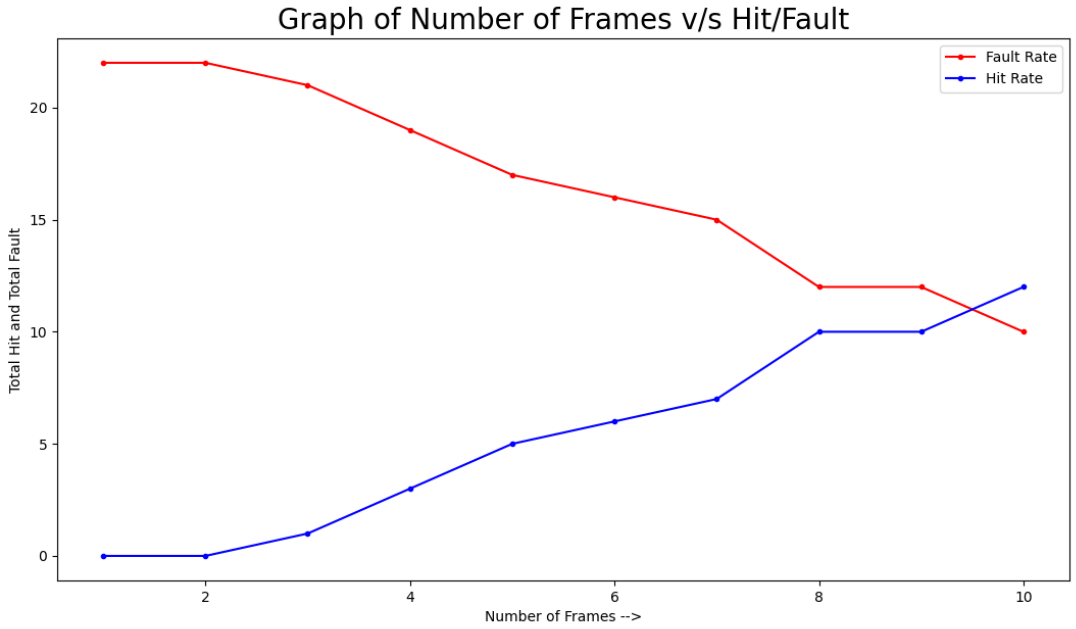
Total Hit: **3**

Total Fault: **19**

Hit Rate: **0.14**

Fault Rate: **0.86**

LRU Graph



Optimal Page Replacement (OPR):

Introduction

Optimal Page Replacement algorithm is the best page replacement algorithm as it gives the least number of page faults. It is also known as OPT, clairvoyant replacement algorithm, or Belady's optimal page replacement policy.

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future, i.e., the pages in the memory which are going to be referred farthest in the future are replaced.

This algorithm was introduced long back and is difficult to implement because it requires future knowledge of the program behavior. However, it is possible to implement optimal page replacement on the second run by using the page reference information collected on the first run.

Advantages

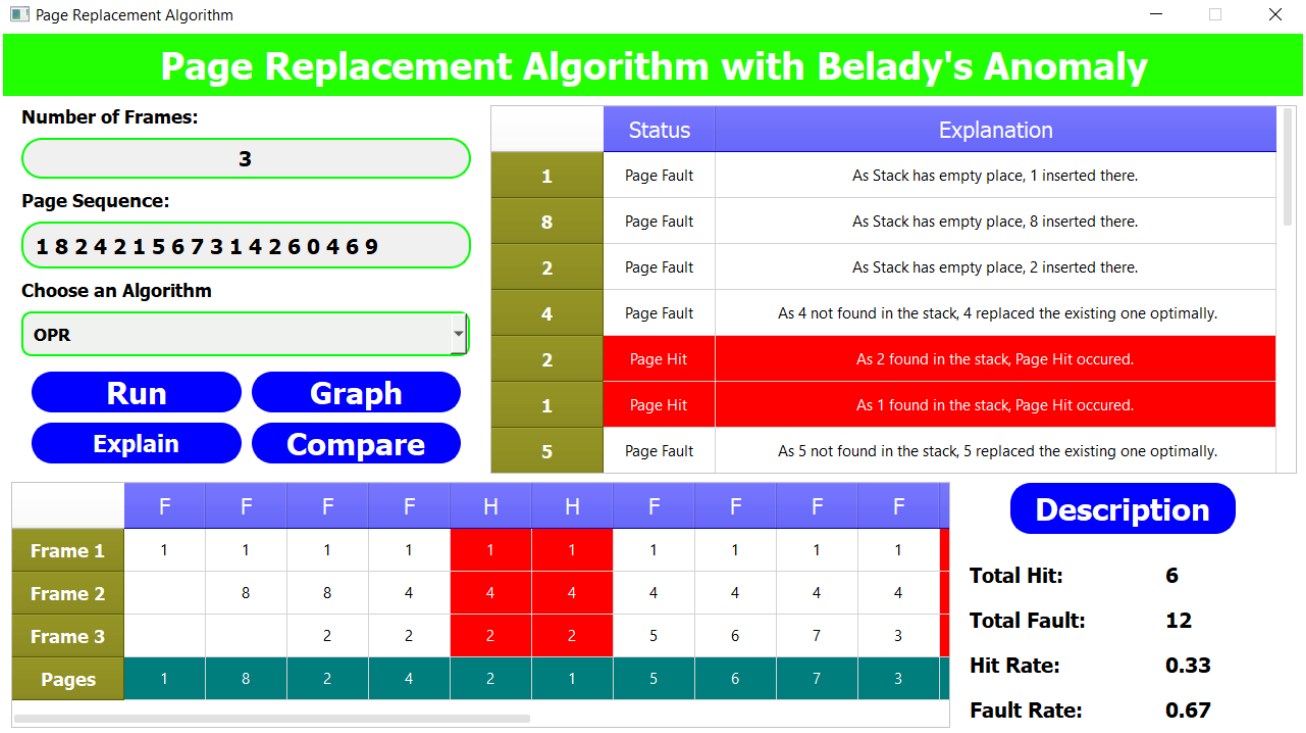
- Easy to implement.
- Simple data structures are used.
- Highly efficient.
- Don't suffer from Belady's Anomaly.

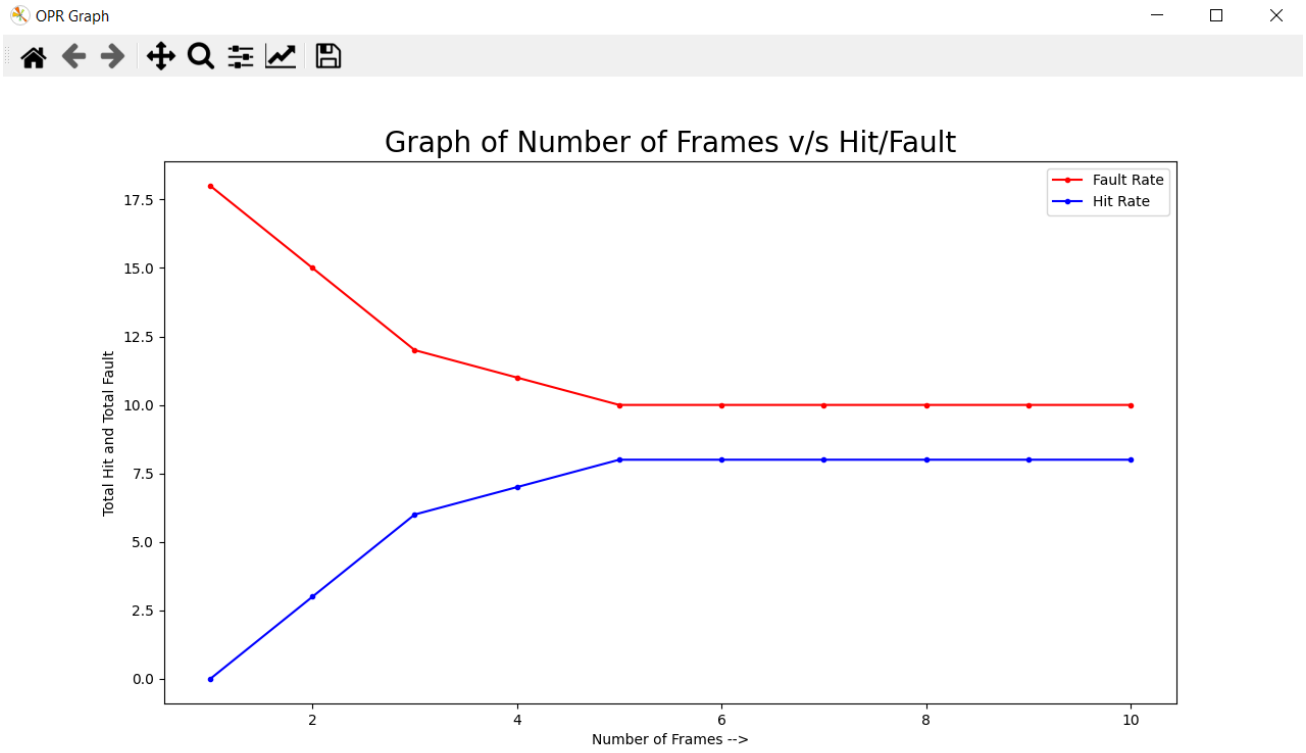
Disadvantages

- Requires future knowledge of the program.
- Time consuming.

Snapshots

Example 1





Example 2

Page Replacement Algorithm

Page Replacement Algorithm with Belady's Anomaly

Number of Frames:

4

Page Sequence:

4 2 1 5 3 4 5 6 2 4 1 5 2 4 8 1 3 0

Choose an Algorithm

OPR

Run | Graph | Explain | Compare

	Status	Explanation
4	Page Fault	As Stack has empty place, 4 inserted there.
2	Page Fault	As Stack has empty place, 2 inserted there.
1	Page Fault	As Stack has empty place, 1 inserted there.
5	Page Fault	As Stack has empty place, 5 inserted there.
3	Page Fault	As 3 not found in the stack, 3 replaced the existing one optimally.
4	Page Hit	As 4 found in the stack, Page Hit occured.
5	Page Hit	As 5 found in the stack, Page Hit occured.

	F	F	F	F	F	H	H	F	H	H
Frame 1	4	4	4	4	4	4	4	4	4	4
Frame 2		2	2	2	2	2	2	2	2	2
Frame 3			1	1	3	3	3	6	6	6
Frame 4				5	5	5	5	5	5	5

Description

Total Hit: 8

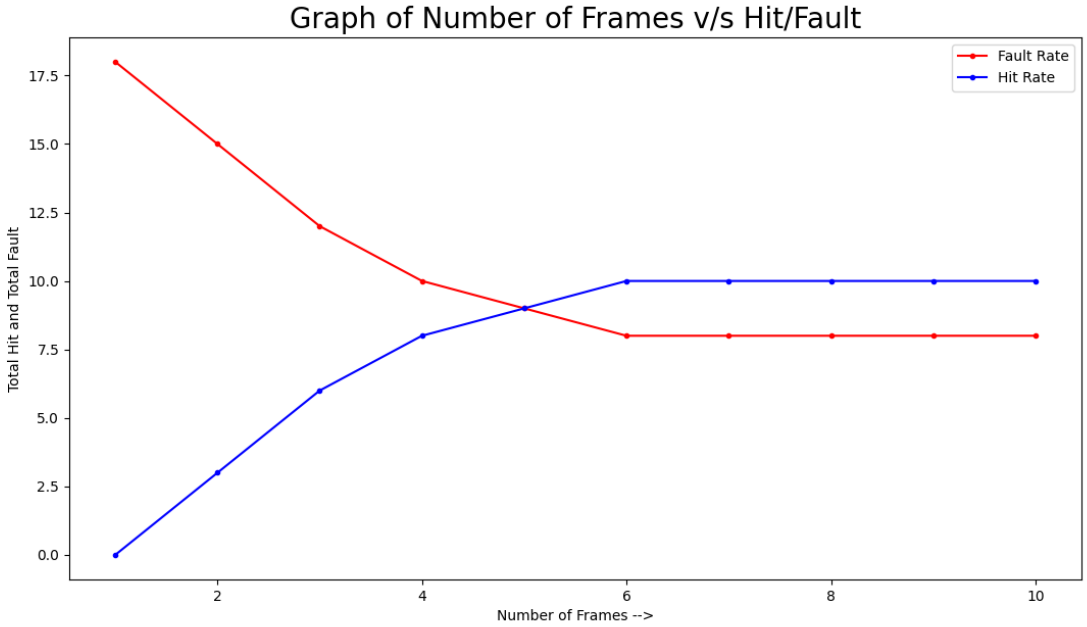
Total Fault: 10

Hit Rate: 0.44

Fault Rate: 0.56

OPR Graph

x=1.917 y=14.53



Random Page Replacement (RPR)

Introduction

Random replacement algorithm replaces a random page in memory. This eliminates the overhead cost of tracking page references. Usually it fares better than FIFO, and for looping memory references it is better than LRU, although generally LRU performs better in practice.

Advantages

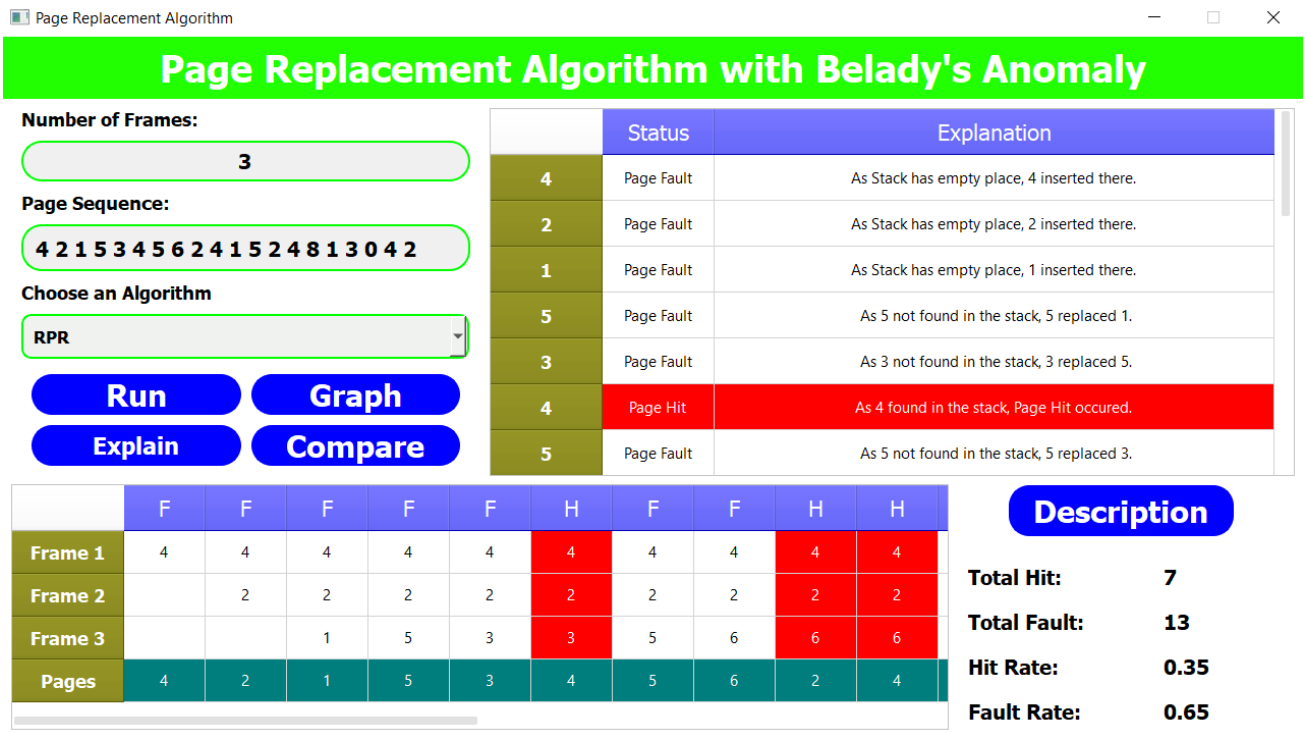
- Easy to implement.
- Simple data structures are used.

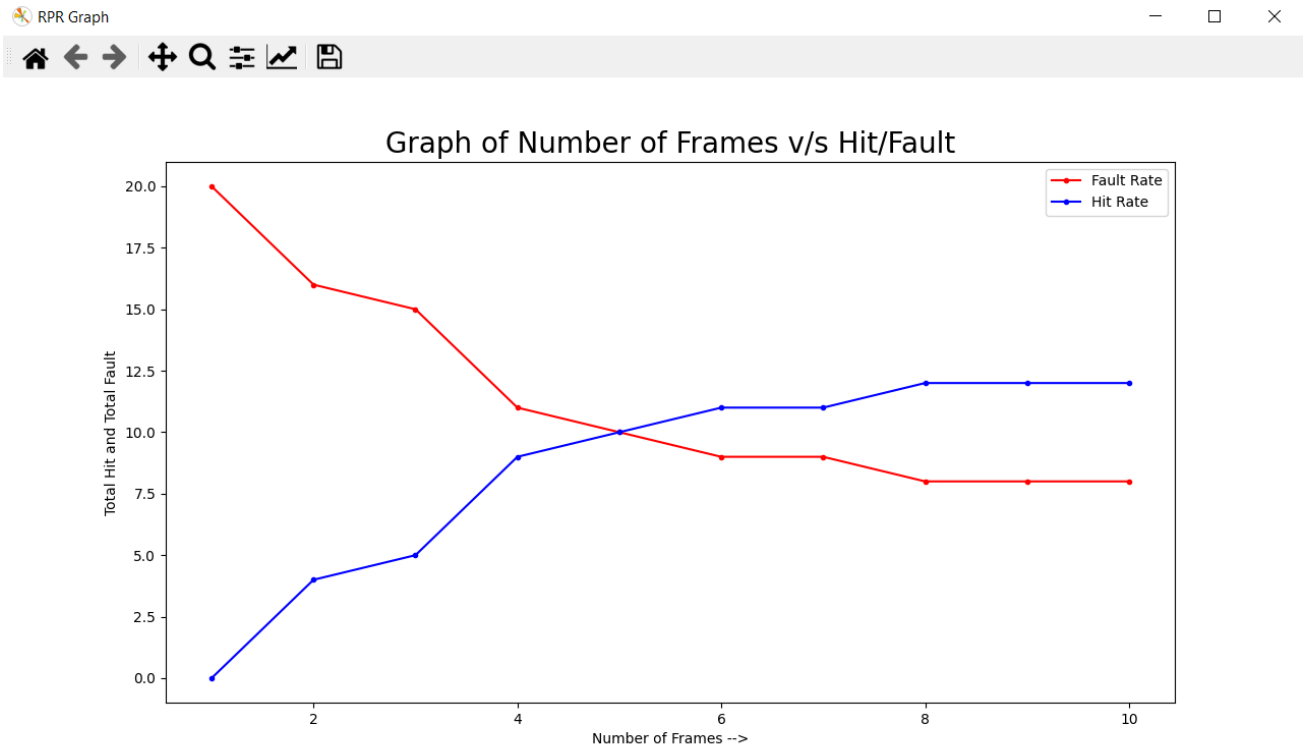
Disadvantages

- We don’t have any control on which page will get replaced.

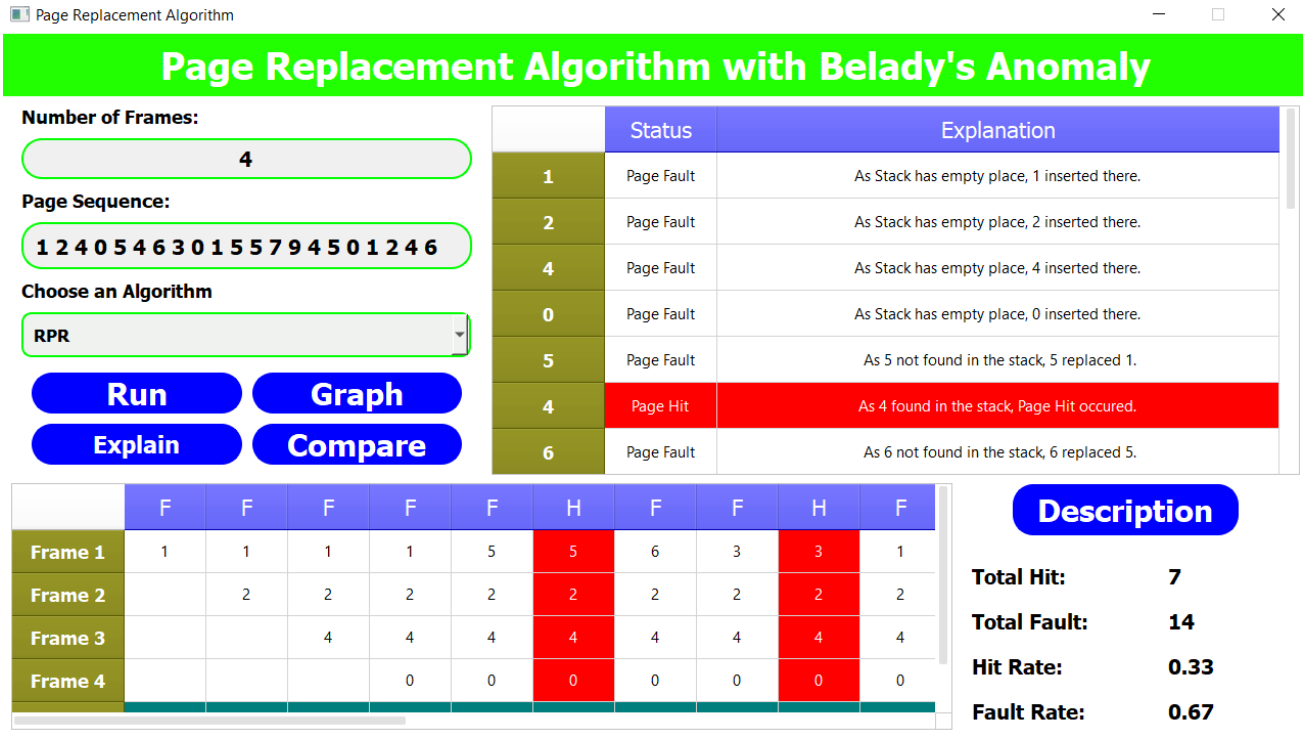
Snapshots

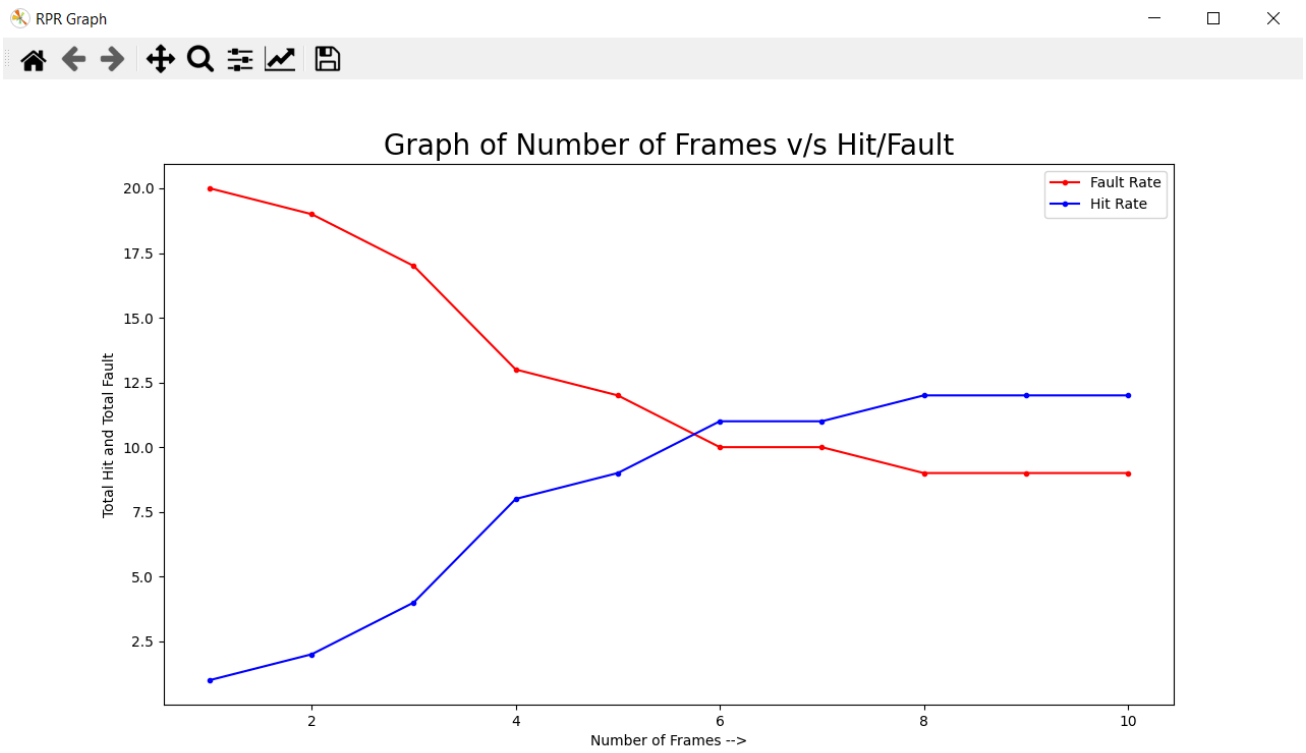
Example 1




RPR Graph
Graph of Number of Frames v/s Hit/Fault
Fault Rate
Hit Rate
Total Hit and Total Fault
Number of Frames -->

Example 2


Page Replacement Algorithm
Page Replacement Algorithm with Belady's Anomaly
Number of Frames:
4
Page Sequence:
1 2 4 0 5 4 6 3 0 1 5 5 7 9 4 5 0 1 2 4 6
Choose an Algorithm
RPR
Run
Graph
Explain
Compare
Status
Explanation
1 Page Fault As Stack has empty place, 1 inserted there.
2 Page Fault As Stack has empty place, 2 inserted there.
4 Page Fault As Stack has empty place, 4 inserted there.
0 Page Fault As Stack has empty place, 0 inserted there.
5 Page Fault As 5 not found in the stack, 5 replaced 1.
4 Page Hit As 4 found in the stack, Page Hit occured.
6 Page Fault As 6 not found in the stack, 6 replaced 5.
F F F F F H F F H F
Frame 1 1 1 1 1 5 5 6 3 3 1
Frame 2 2 2 2 2 2 2 2 2 2
Frame 3 4 4 4 4 4 4 4 4
Frame 4 0 0 0 0 0 0 0 0
Description
Total Hit: 7
Total Fault: 14
Hit Rate: 0.33
Fault Rate: 0.67



Second Chance Page Replacement (SCPR)

Introduction

It can be implemented by adding a "second chance" bit to each memory frame-every time the frame is considered (due to a reference made to the page inside it), this bit is set to 1, which gives the page a second chance, as when we consider the candidate page for replacement, we replace the first one with this bit set to 0 (while zeroing out bits of the other pages we see in the process). Thus, a page with the "second chance" bit set to 1 is never replaced during the first consideration and will only be replaced if all the other pages deserve a second chance too!

Advantages

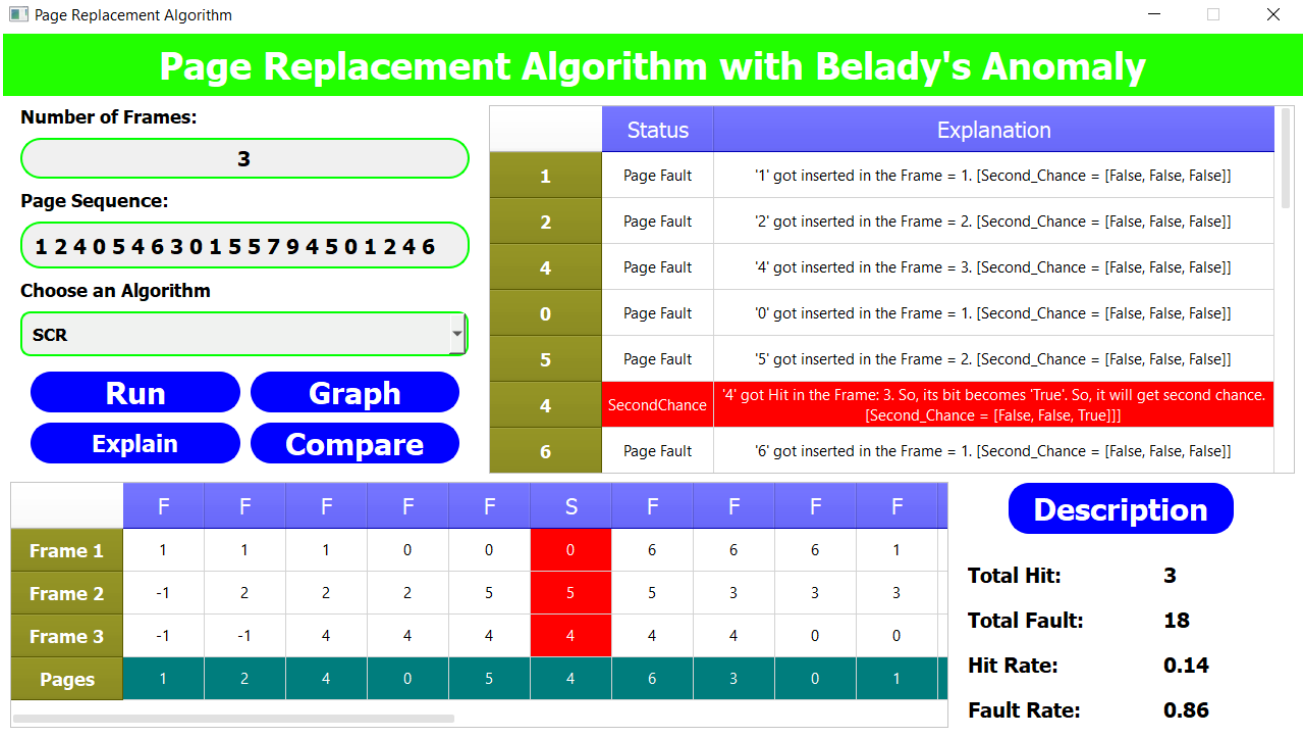
- Highly efficient.
- Simple data structures are used.

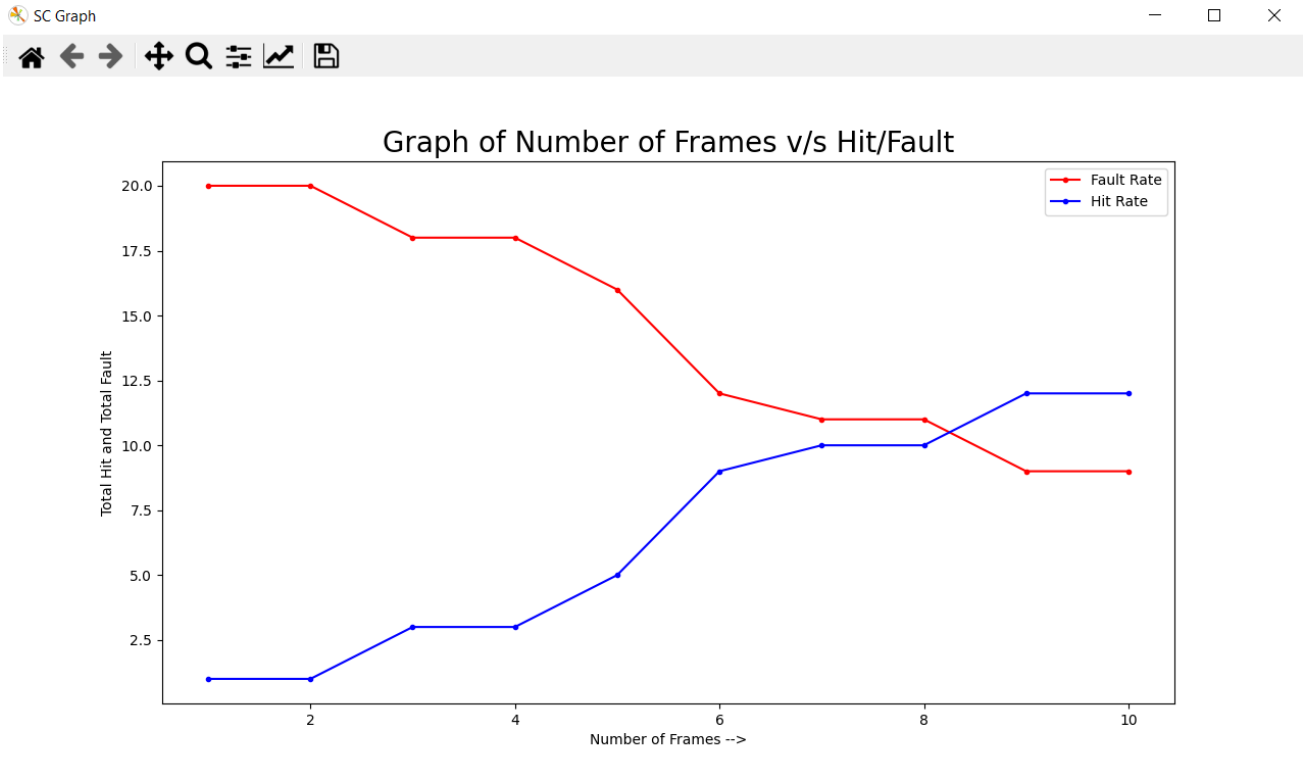
Disadvantages

- Time consuming.

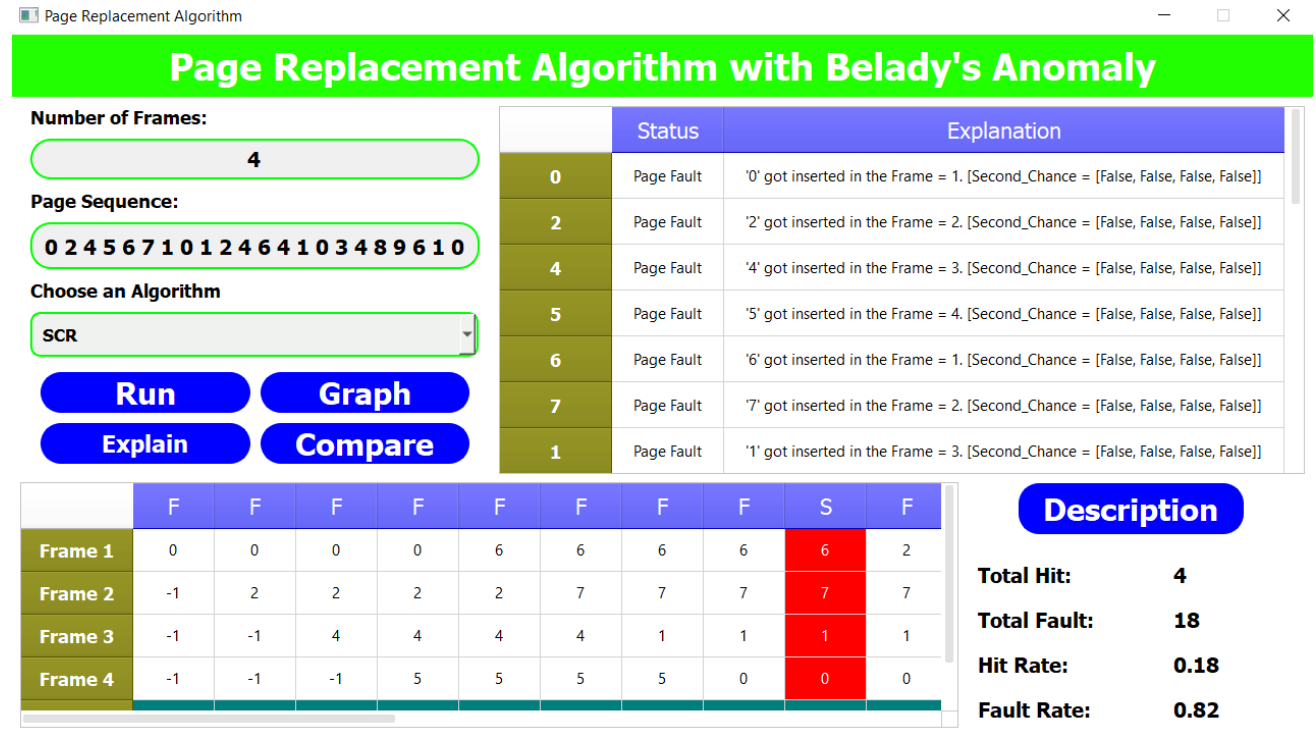
Snapshots

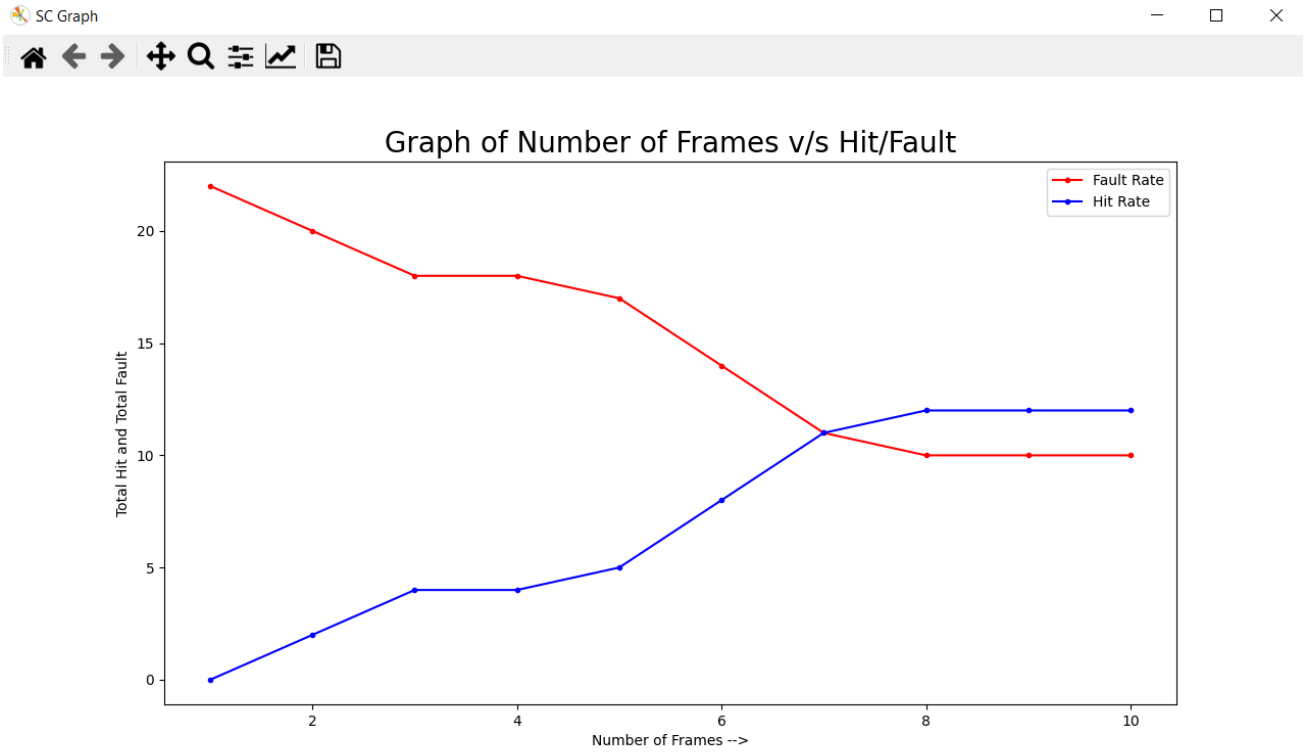
Example 1





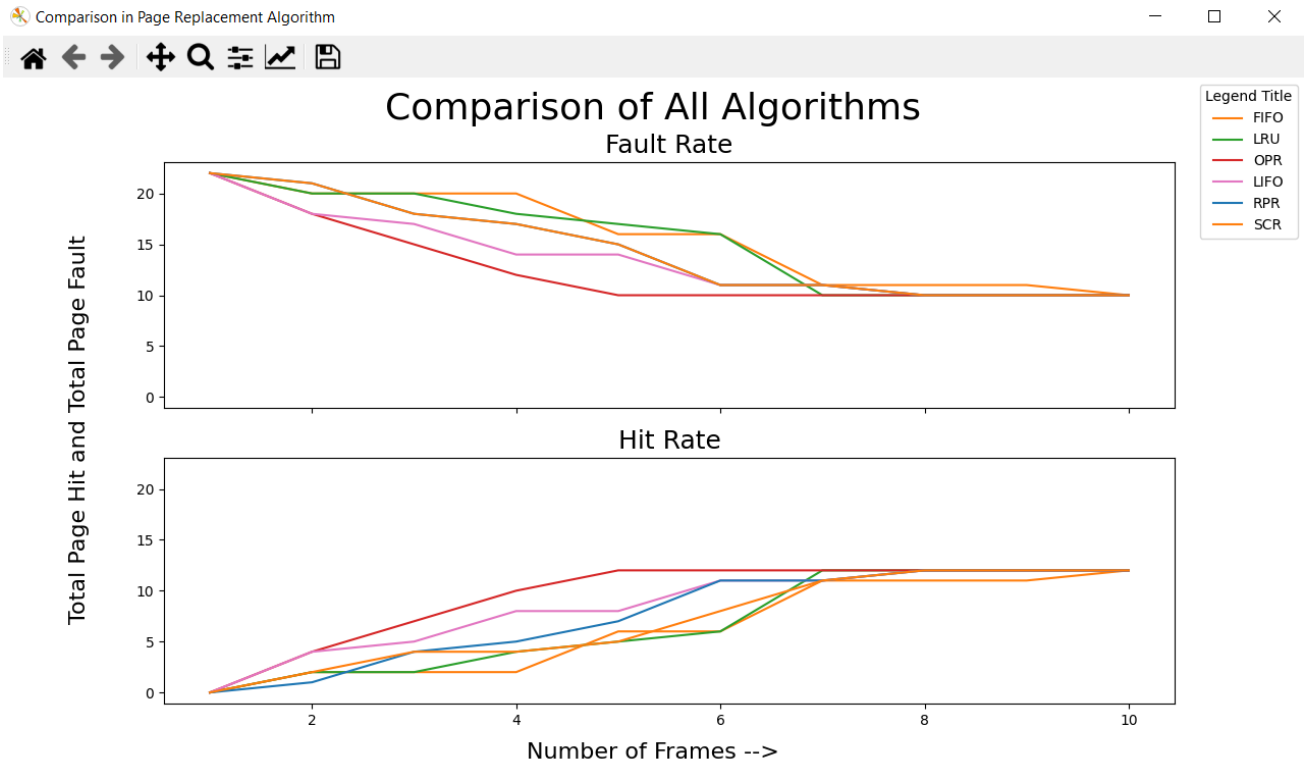
Example 2


Page Replacement Algorithm
Page Replacement Algorithm with Belady's Anomaly
Number of Frames:
4
Page Sequence:
0 2 4 5 6 7 1 0 1 2 4 6 4 1 0 3 4 8 9 6 1 0
Choose an Algorithm
SCR
Run
Graph
Explain
Compare
Status
Explanation
0 Page Fault '0' got inserted in the Frame = 1. [Second_Chance = [False, False, False, False]]
2 Page Fault '2' got inserted in the Frame = 2. [Second_Chance = [False, False, False, False]]
4 Page Fault '4' got inserted in the Frame = 3. [Second_Chance = [False, False, False, False]]
5 Page Fault '5' got inserted in the Frame = 4. [Second_Chance = [False, False, False, False]]
6 Page Fault '6' got inserted in the Frame = 1. [Second_Chance = [False, False, False, False]]
7 Page Fault '7' got inserted in the Frame = 2. [Second_Chance = [False, False, False, False]]
1 Page Fault '1' got inserted in the Frame = 3. [Second_Chance = [False, False, False, False]]
F F F F F F F F S F
Frame 1 0 0 0 0 6 6 6 6 6 2
Frame 2 -1 2 2 2 2 7 7 7 7 7
Frame 3 -1 -1 4 4 4 4 1 1 1 1
Frame 4 -1 -1 -1 5 5 5 5 0 0 0
Description
Total Hit: 4
Total Fault: 18
Hit Rate: 0.18
Fault Rate: 0.82

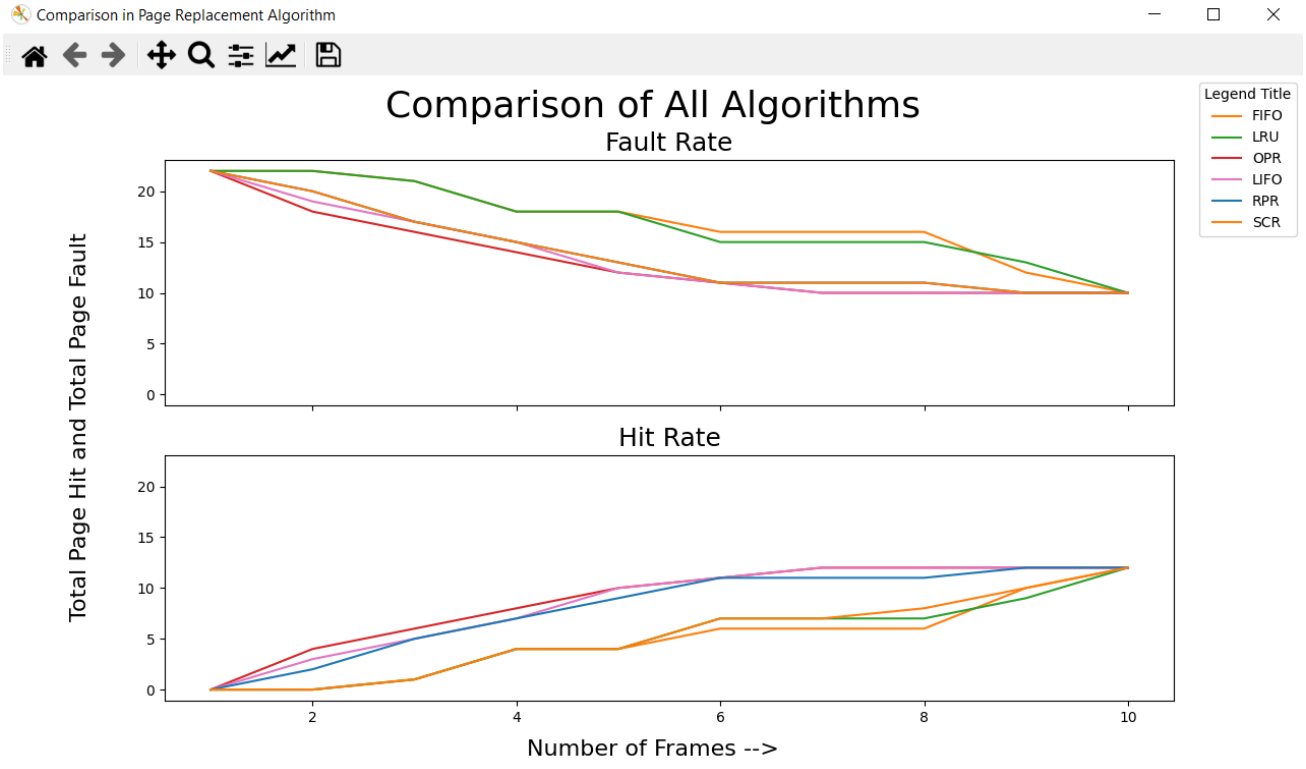

SC Graph
Graph of Number of Frames v/s Hit/Fault
Fault Rate
Hit Rate
Total Hit and Total Fault
Number of Frames -->

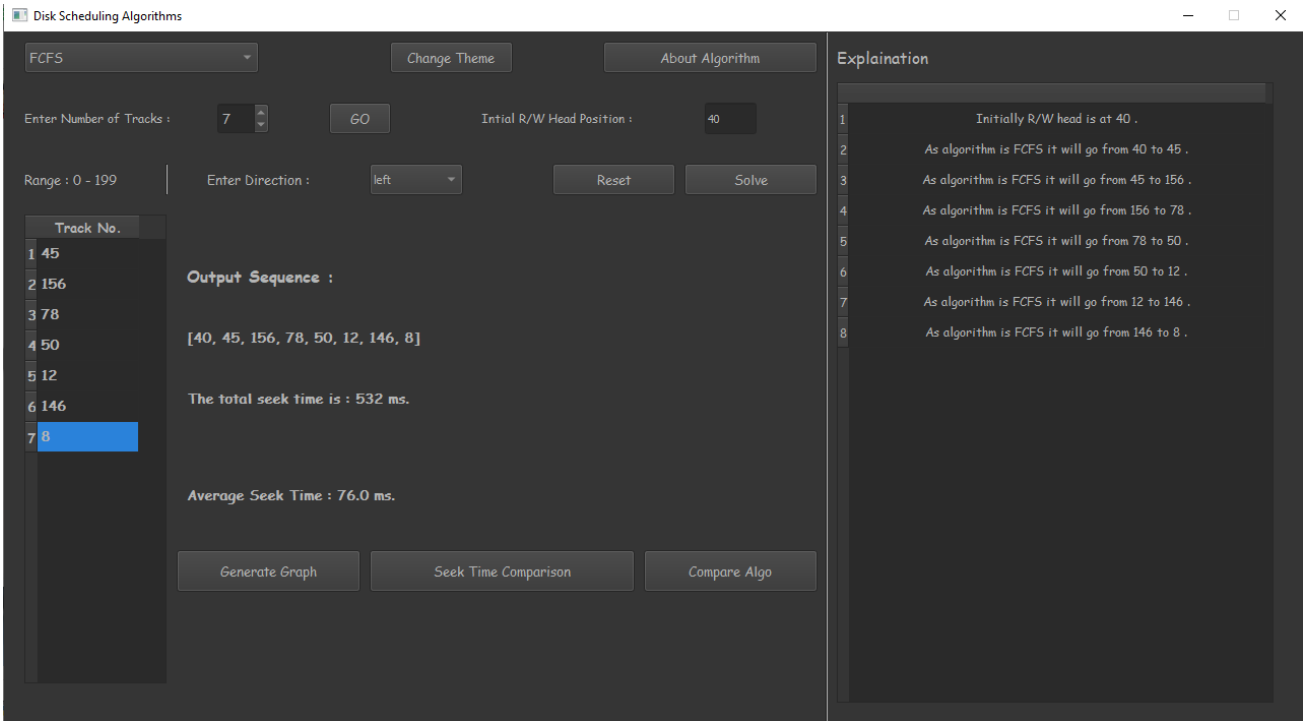
Comparison of All Algorithms

Input String: 0 2 4 5 6 7 1 0 1 2 4 6 4 1 0 3 4 8 9 6 1 0



Input String: 4 5 2 1 0 3 4 5 7 8 9 6 4 1 0 2 4 1 2 0 3 5





Disk Scheduler

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.
Disk scheduling is important because:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

Seek Time

Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So, the disk scheduling algorithm that gives minimum average seek time is better.

Rotation Latency

- Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So, the disk scheduling algorithm that gives minimum rotational latency is better.

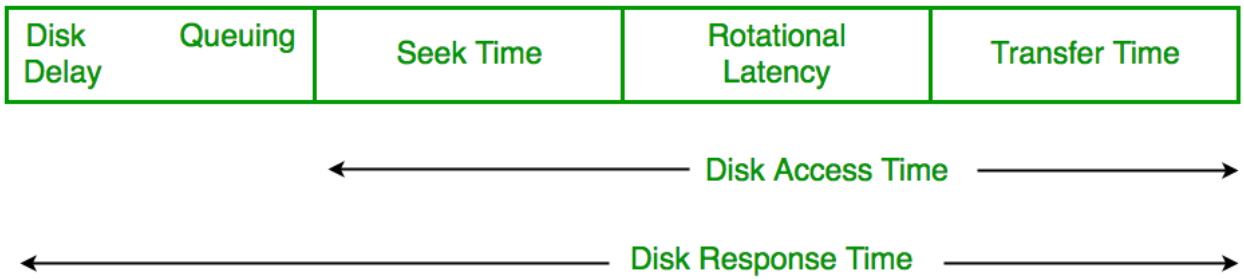
Transfer Time

- Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

Disk Access Time

Disk access time is given as,

Disk Access Time = Rotational Latency + Seek Time + Transfer Time



Disk Response Time

Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of the all requests. Variance Response Time is measure of how individual request are serviced with respect to average response time. So, the disk scheduling algorithm that gives minimum variance response time is better.

Purpose of Disk Scheduling

The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

Goal of Disk Scheduling Algorithm

- Fairness
- High throughout
- Minimal traveling head time

Disk Scheduling Algorithms

The list of various disks scheduling algorithm is given below. Each algorithm is carrying some advantages and disadvantages. The limitation of each algorithm leads to the evolution of a new algorithm.

- FCFS scheduling algorithm
- SSTF (shortest seek time first) algorithm
- SCAN scheduling
- C-SCAN scheduling
- LOOK Scheduling
- C-LOOK scheduling

First Come First Serve (FCFS)

Introduction

FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.

Advantages

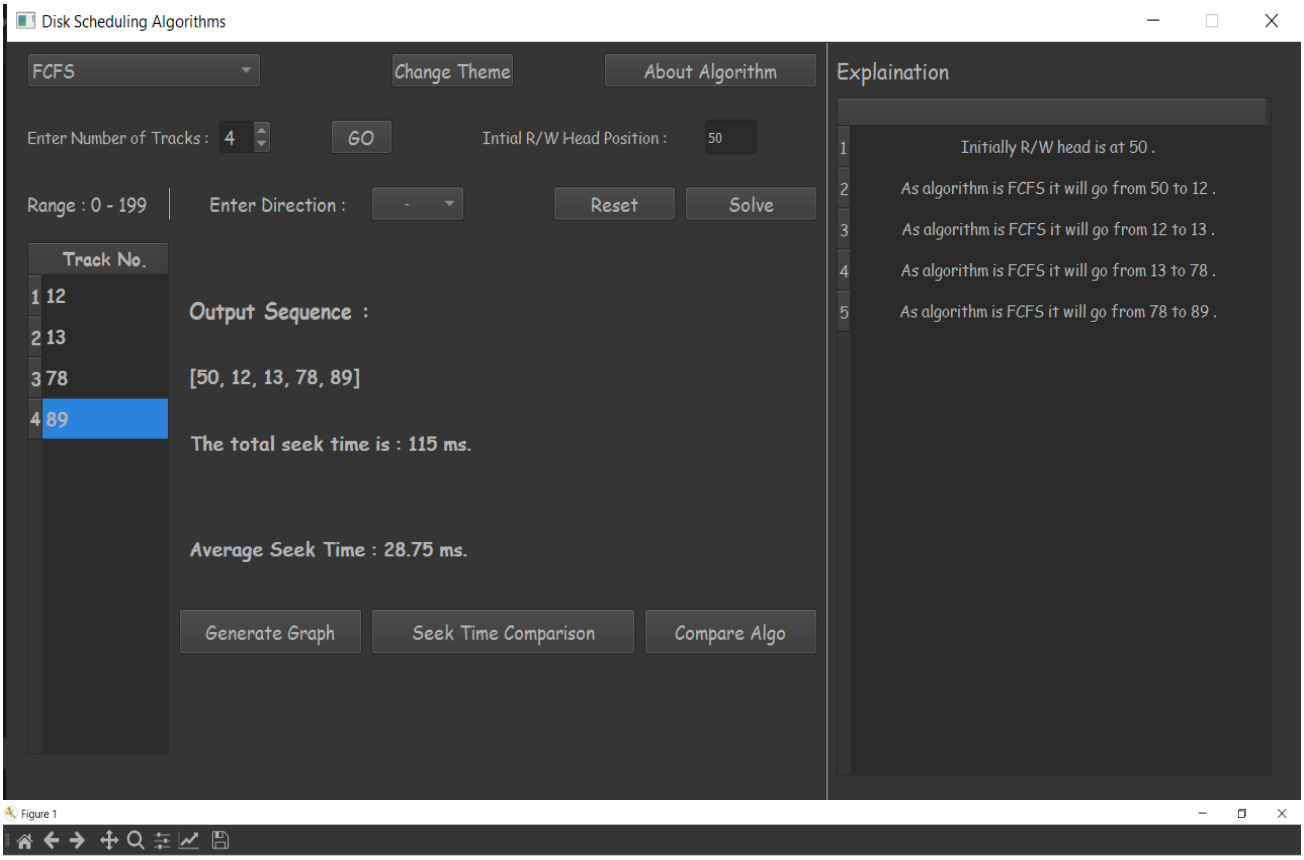
- First Come First Serve algorithm has a very simple logic, it executes the process requests one by one in the sequence they arrive.
- Thus, First Come First Serve is very simple and easy to understand and implement.
- In FCFS eventually, each and every process gets a chance to execute, so no starvation occurs.

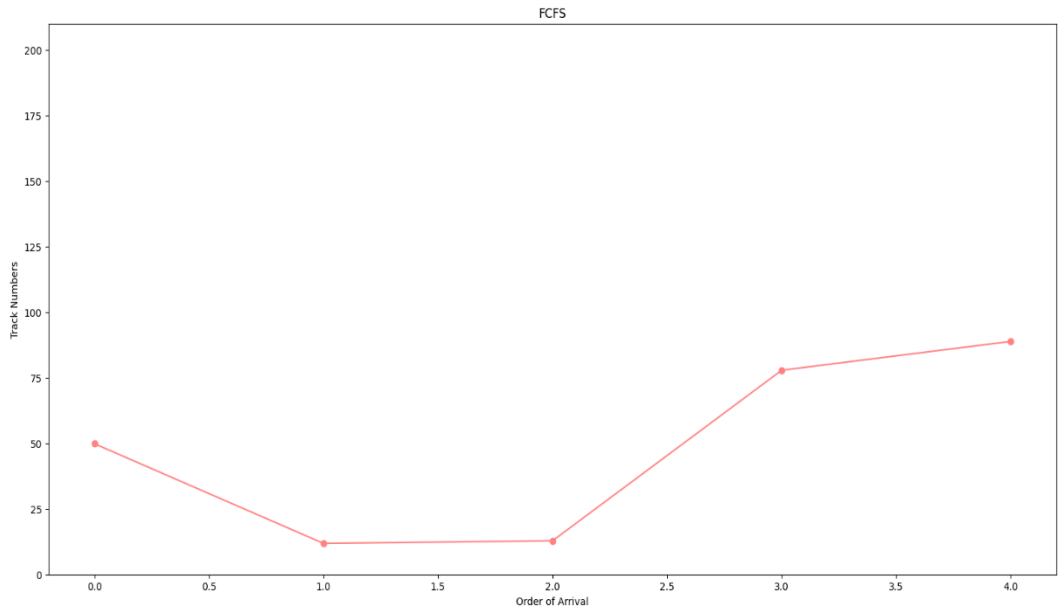
Disadvantages

- This scheduling algorithm is non-pre-emptive, which means the process can't be stopped in middle of execution and will run its full course.
- FCFS being a non-pre-emptive scheduling algorithm, the short processes which are at the back of the queue have to wait for the long process at the front to finish
- The throughput of FCFS is not very efficient.
- FCFS is implemented on small systems only where input-output efficiency is not of utmost importance.

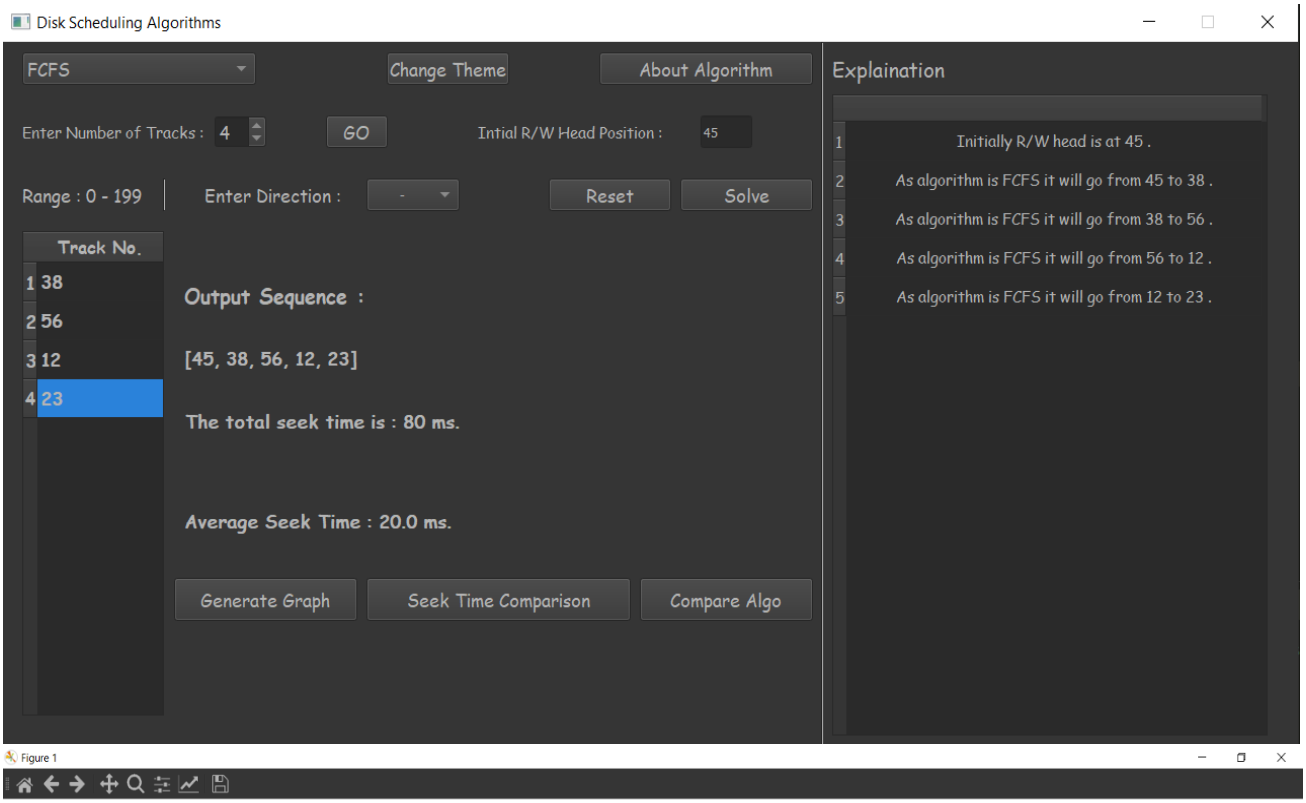
Snapshots

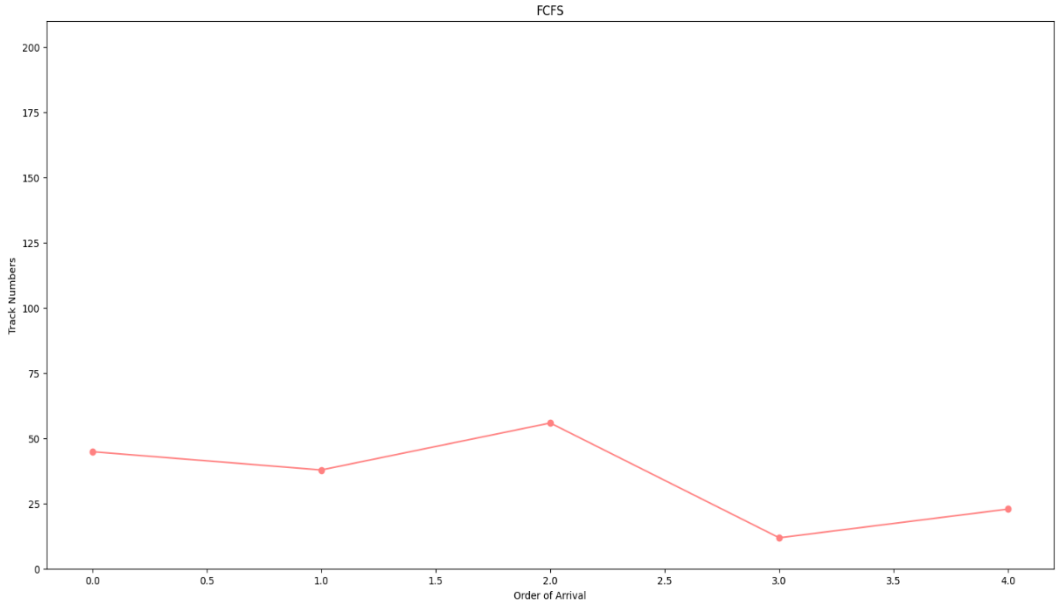
Example 1


Disk Scheduling Algorithms
FCFS
Change Theme
About Algorithm
Enter Number of Tracks : 4
GO
Intial R/W Head Position : 50
Range : 0 - 199
Enter Direction :
Reset
Solve
Track No.
1 12
2 13
3 78
4 89
Output Sequence :
[50, 12, 13, 78, 89]
The total seek time is : 115 ms.
Average Seek Time : 28.75 ms.
Generate Graph
Seek Time Comparison
Compare Algo
Explaination
1 Initially R/W head is at 50 .
2 As algorithm is FCFS it will go from 50 to 12 .
3 As algorithm is FCFS it will go from 12 to 13 .
4 As algorithm is FCFS it will go from 13 to 78 .
5 As algorithm is FCFS it will go from 78 to 89 .
Figure 1


FCFS
Track Numbers
Order of Arrival
0
25
50
75
100
125
150
175
200
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Example 2





Shortest Seek Time First (SSTF)

Introduction

In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed

first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

Advantages

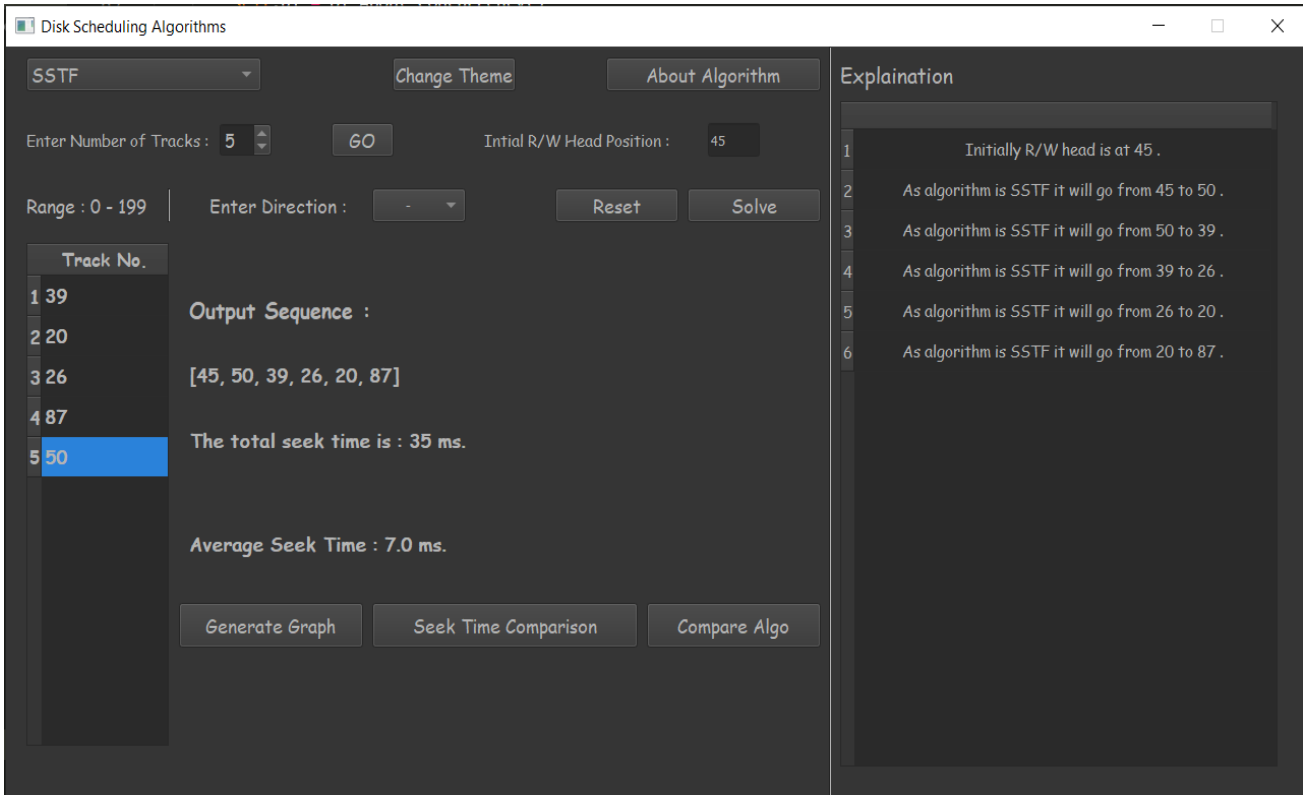
- The total seek time is reduced compared to First Come First Serve.
- SSTF improves and increases throughput.
- Less average waiting time and response time in SSTF.

Disadvantages

- In SSTF there is an overhead of finding out the closest request.
- Starvation may occur for requests far from head.
- In SSTF high variance is present in response time and waiting time.
- Frequent switching of the Head's direction slows the algorithm.

Snapshots

Example 1



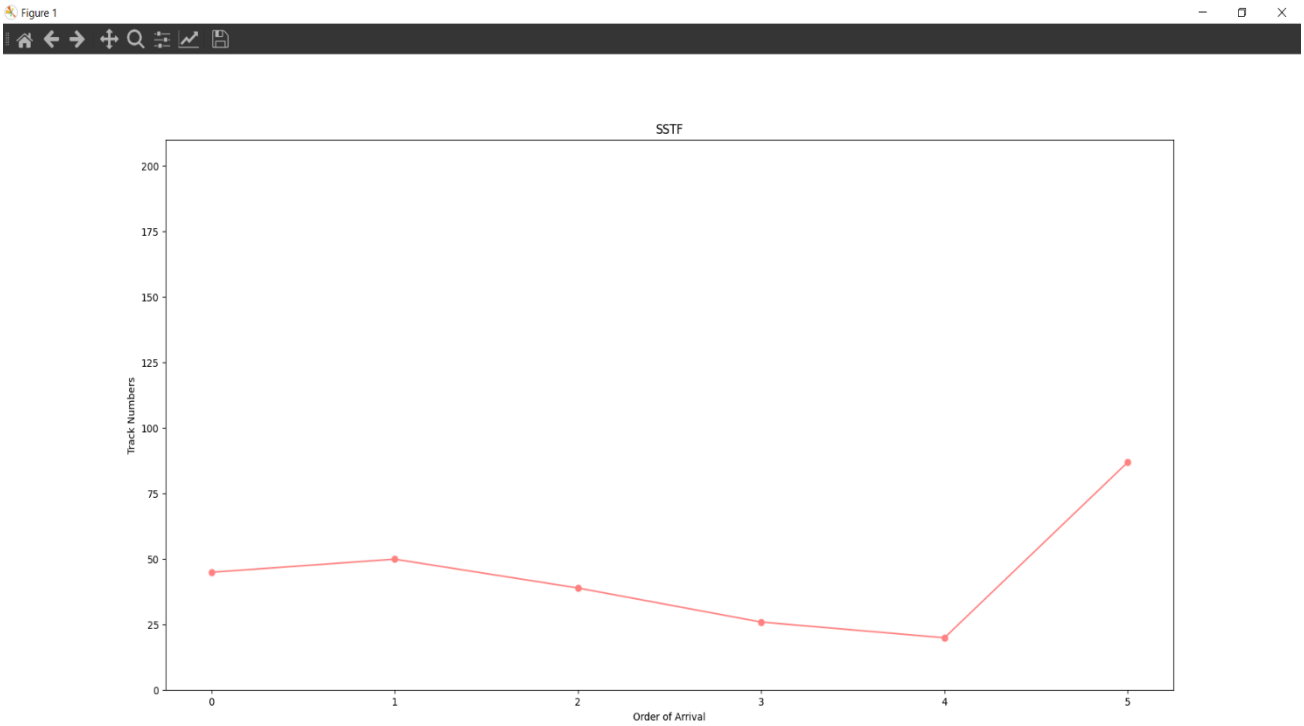
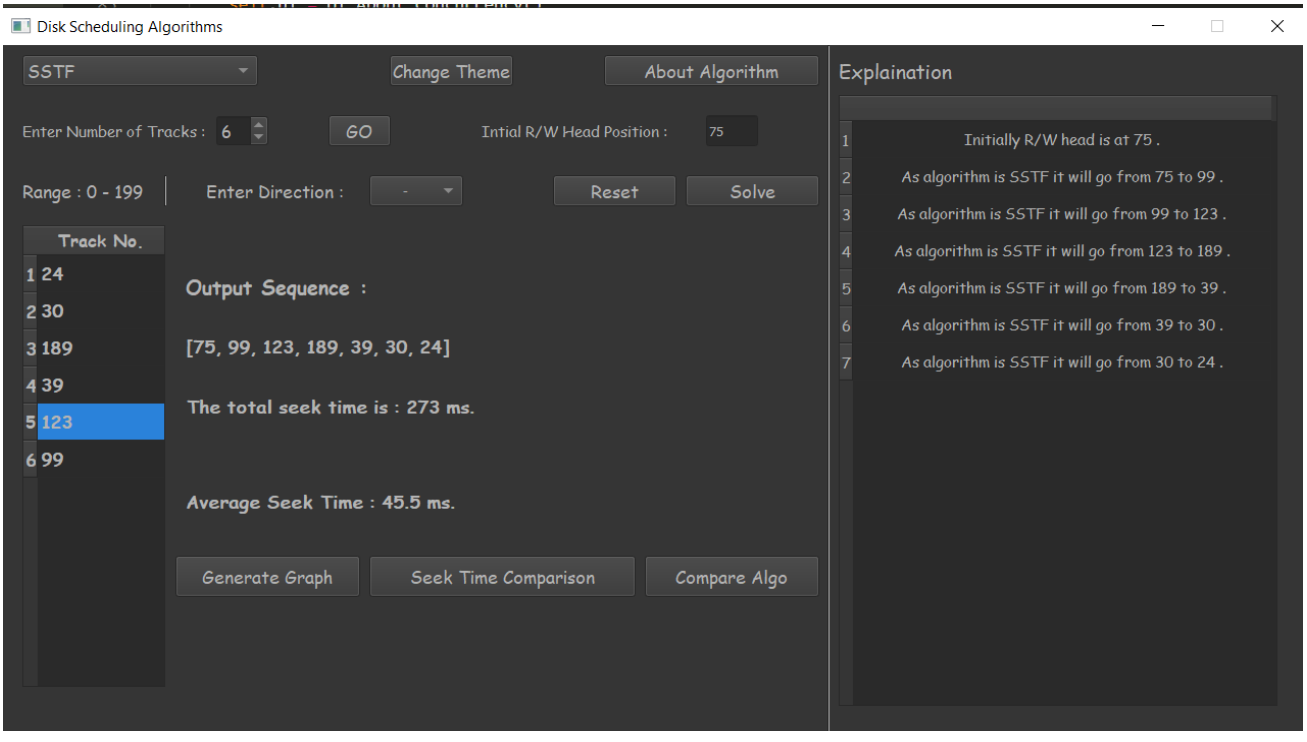
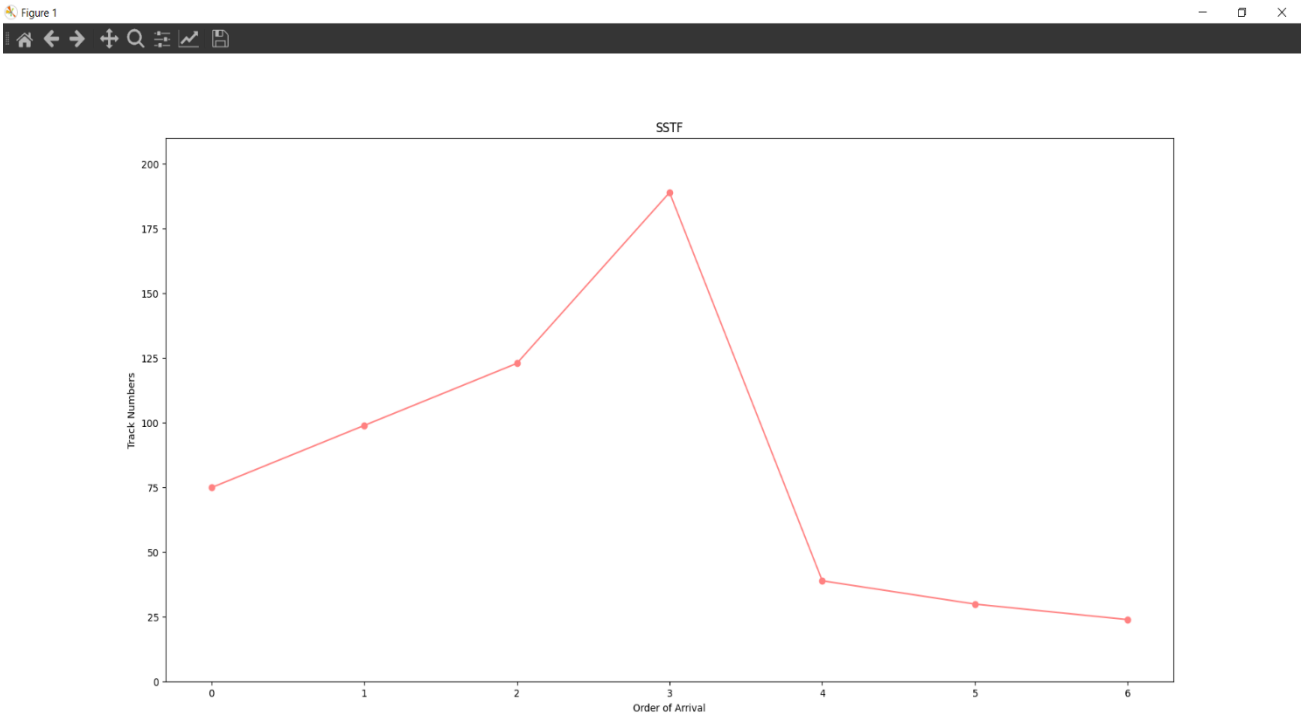


Figure 1
SSTF
Track Numbers
Order of Arrival
200
175
150
125
100
75
50
25
0
0
1
2
3
4
5

Example 2


Disk Scheduling Algorithms
SSTF
Change Theme
About Algorithm
Enter Number of Tracks : 6
GO
Intial R/W Head Position : 75
Range : 0 - 199
Enter Direction :
Reset
Solve
Track No.
1 24
2 30
3 189
4 39
5 123
6 99
Output Sequence :
[75, 99, 123, 189, 39, 30, 24]
The total seek time is : 273 ms.
Average Seek Time : 45.5 ms.
Generate Graph
Seek Time Comparison
Compare Algo
Explaination
1 Initially R/W head is at 75 .
2 As algorithm is SSTF it will go from 75 to 99 .
3 As algorithm is SSTF it will go from 99 to 123 .
4 As algorithm is SSTF it will go from 123 to 189 .
5 As algorithm is SSTF it will go from 189 to 39 .
6 As algorithm is SSTF it will go from 39 to 30 .
7 As algorithm is SSTF it will go from 30 to 24 .



SCAN

Introduction

In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and hence also known as **elevator algorithm.** As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Advantages

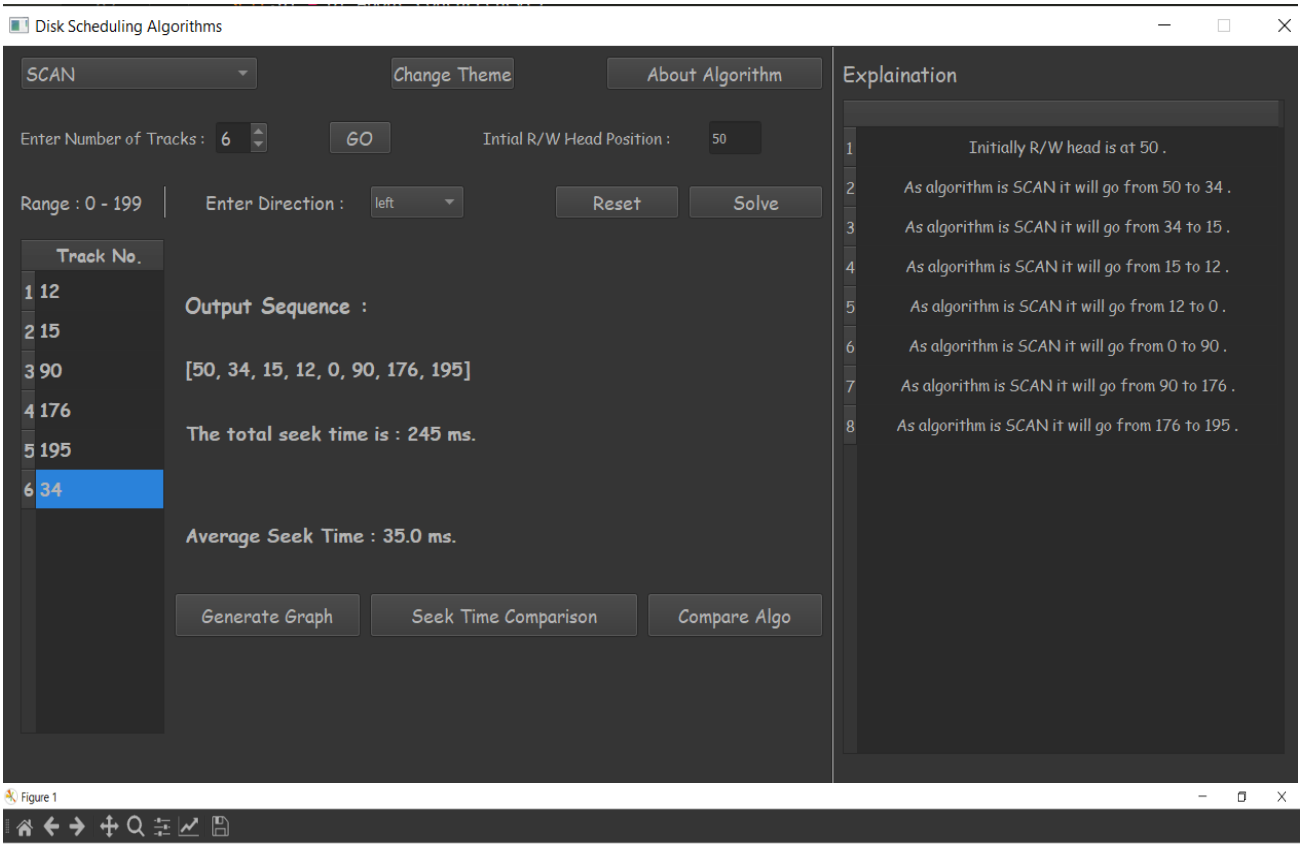
- Scan scheduling algorithm is simple and easy to understand and implement.
- Starvation is avoided in SCAN algorithm.
- Low variance Occurs in waiting time and response time.

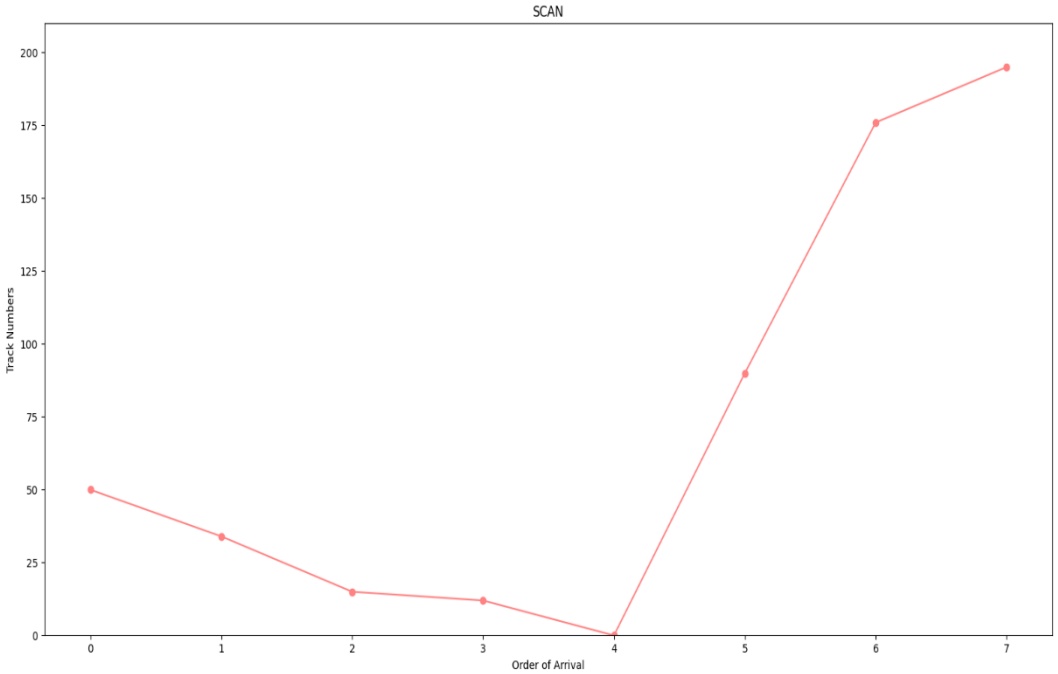
Disadvantages

- Long waiting time occurs for the cylinders which are just visited by the head.
- In SCAN the head moves till the end of the disk despite the absence of requests to be serviced.

Snapshots

Example 1





Example 2

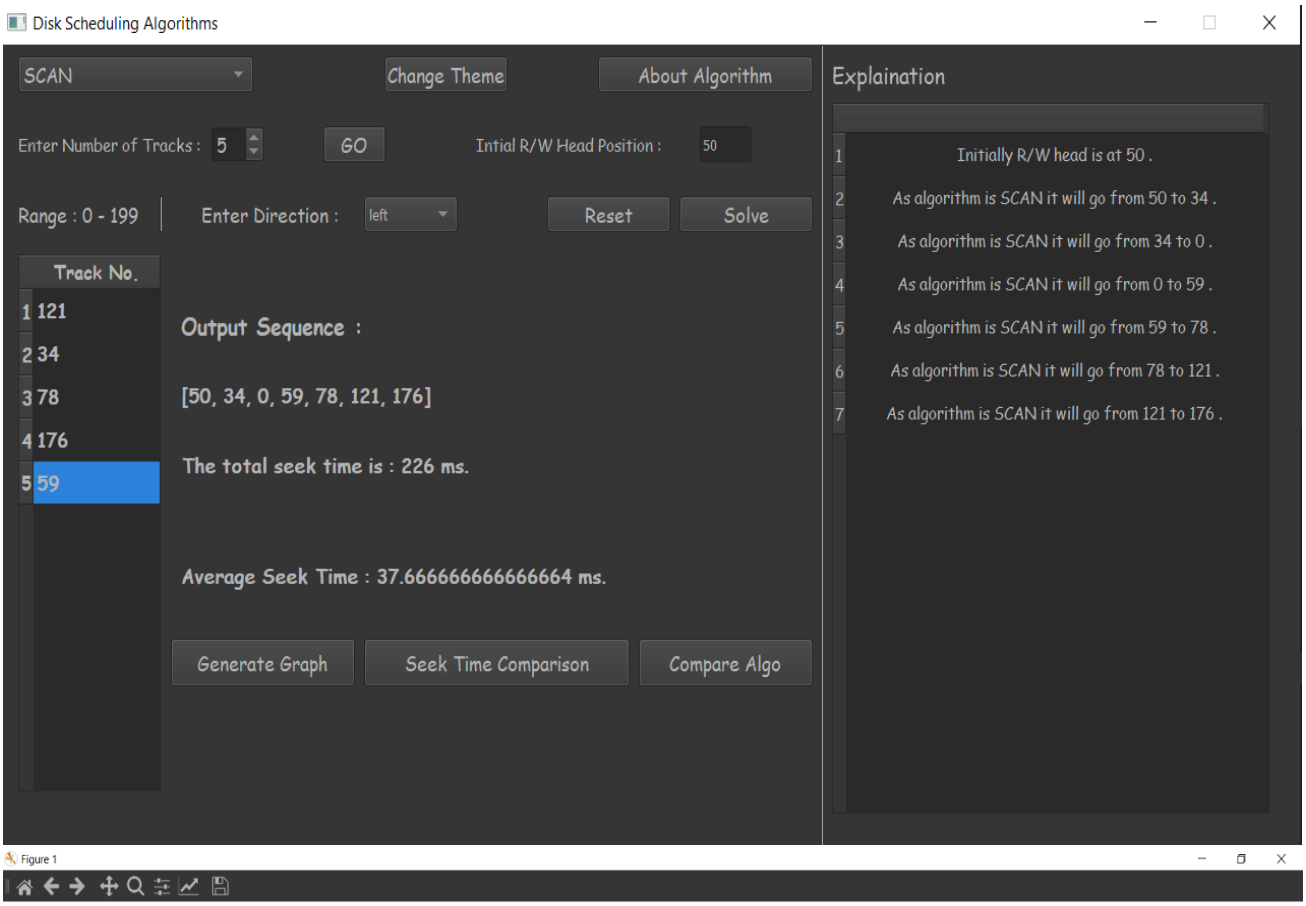
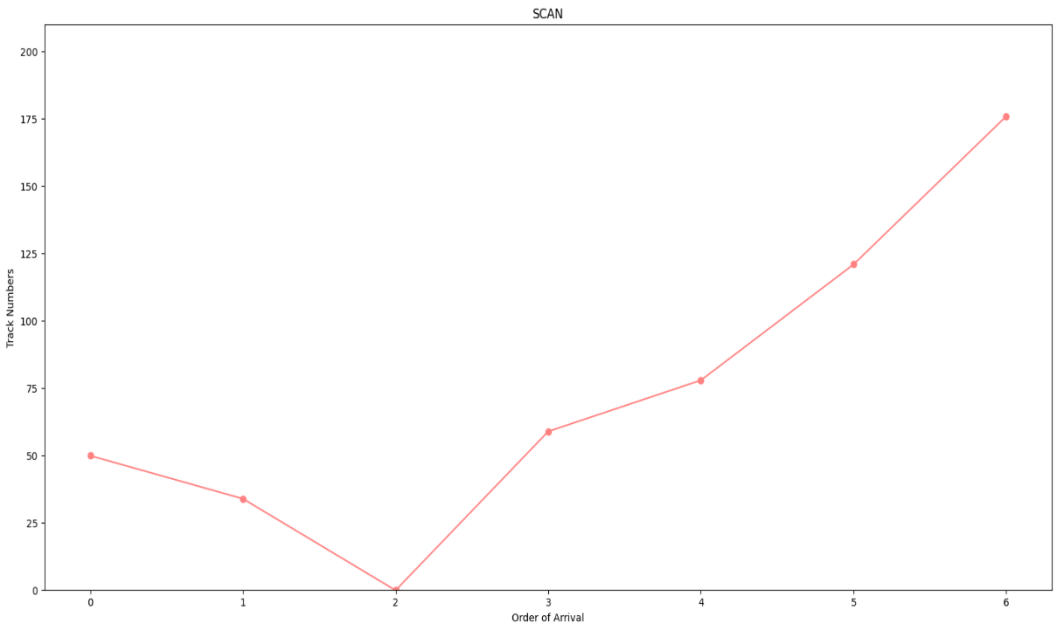

Disk Scheduling Algorithms
SCAN
Change Theme
About Algorithm
Enter Number of Tracks : 5
GO
Intial R/W Head Position : 50
Range : 0 - 199
Enter Direction : left
Reset
Solve
Track No.
1 121
2 34
3 78
4 176
5 59
Output Sequence :
[50, 34, 0, 59, 78, 121, 176]
The total seek time is : 226 ms.
Average Seek Time : 37.666666666666664 ms.
Generate Graph
Seek Time Comparison
Compare Algo
Explaination
1 Initially R/W head is at 50 .
2 As algorithm is SCAN it will go from 50 to 34 .
3 As algorithm is SCAN it will go from 34 to 0 .
4 As algorithm is SCAN it will go from 0 to 59 .
5 As algorithm is SCAN it will go from 59 to 78 .
6 As algorithm is SCAN it will go from 78 to 121 .
7 As algorithm is SCAN it will go from 121 to 176 .

Figure 1


SCAN
Track Numbers
Order of Arrival
200
175
150
125
100
75
50
25
0
0
1
2
3
4
5
6

C-SCAN

Introduction

In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in CSCAN algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Advantages

- C-SCAN Algorithm is the successor and the improved version of the SCAN scheduling Algorithm.
- The Head move from one end to the other of the disk while serving all the requests in between.
- The waiting time for the cylinders which were just visited by the head is reduced in C-SCAN compared to the SCAN Algorithm.
- Uniform waiting time is provided.
- Better response time is provided.

Disadvantages

- More seek movements are caused in C-SCAN compared to SCAN Algorithm.
- In C-SCAN even if there are no requests left to be serviced the Head will still travel to the end of the disk unlike SCAN algorithm.

Snapshots

Example 1

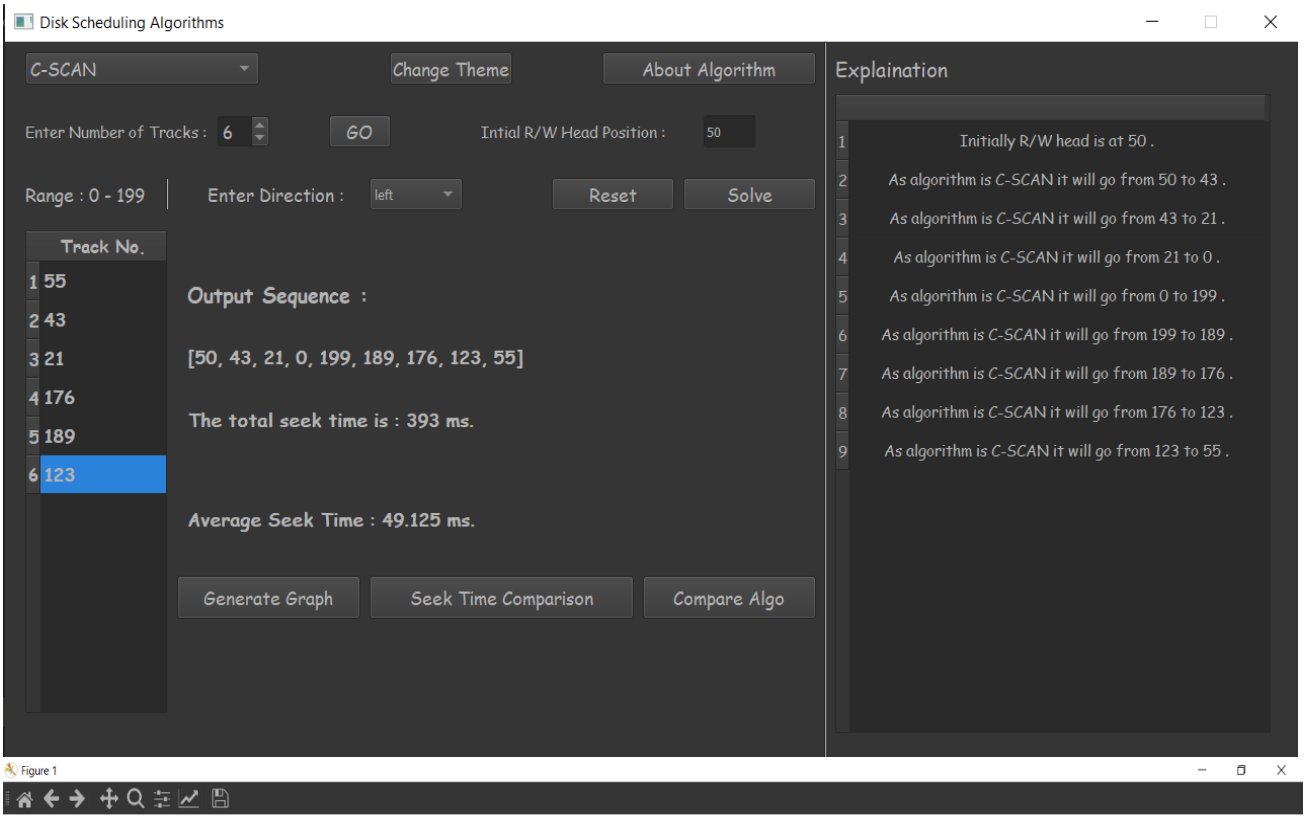
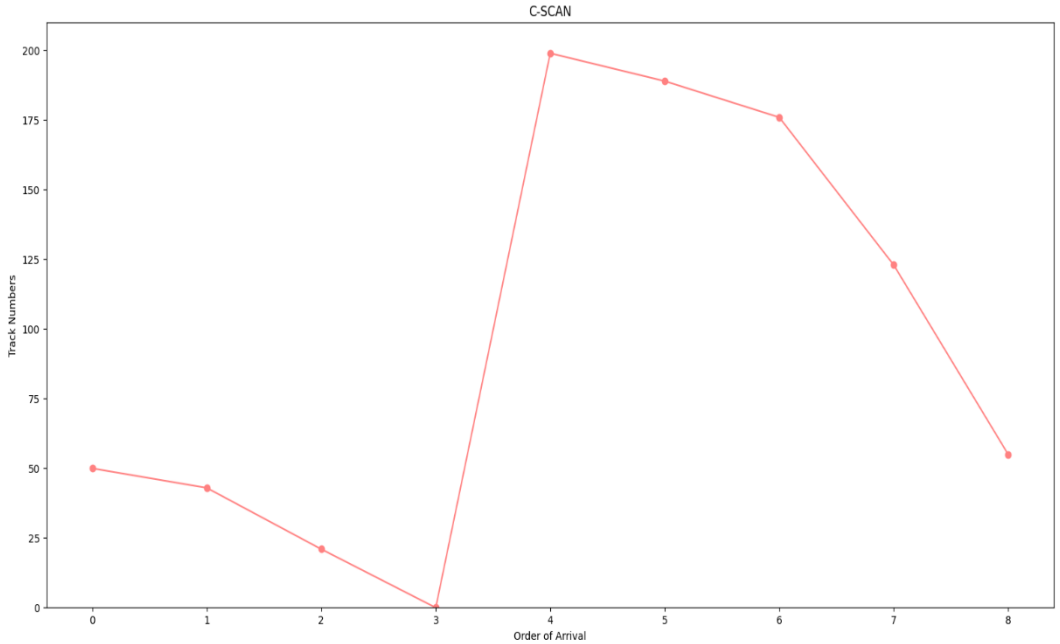
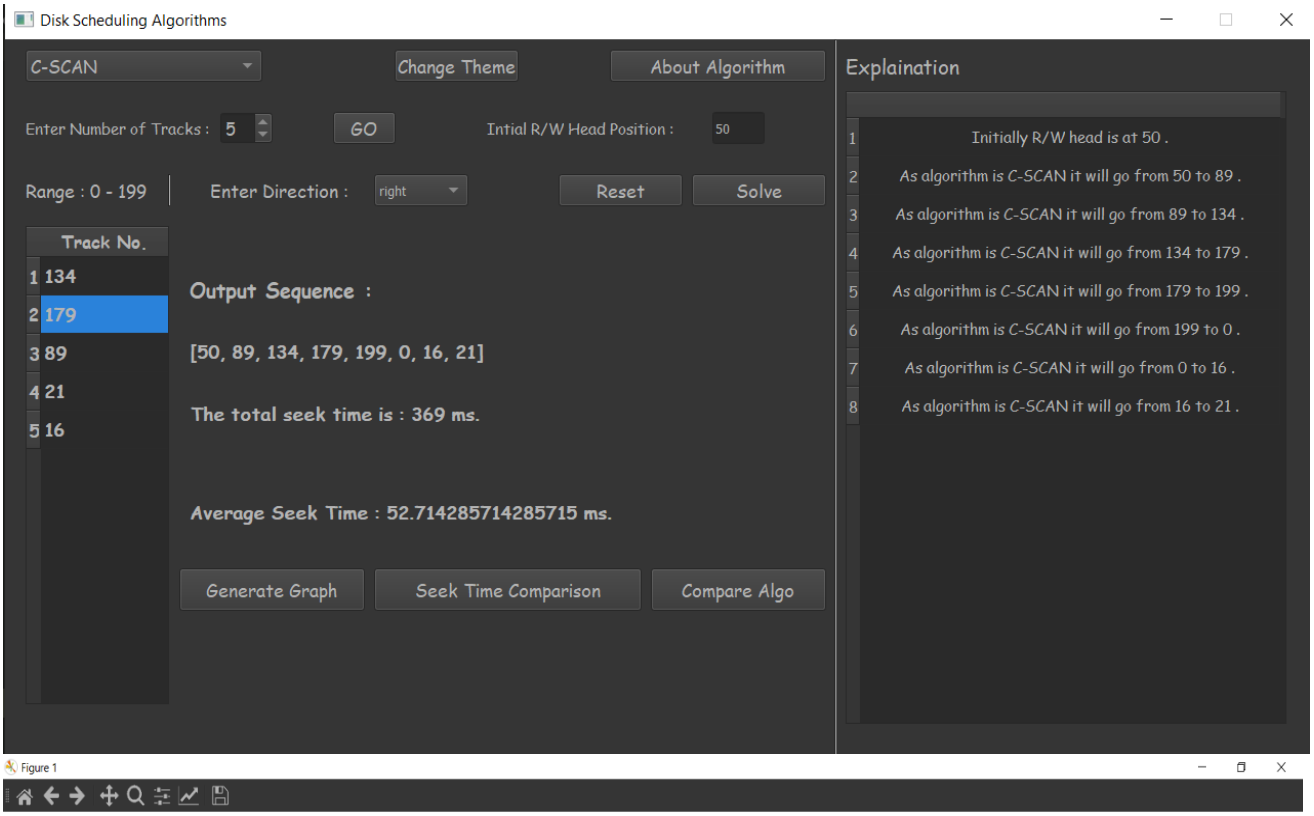

Disk Scheduling Algorithms
C-SCAN
Change Theme
About Algorithm
Enter Number of Tracks : 6
GO
Intial R/W Head Position : 50
Range : 0 - 199
Enter Direction : left
Reset
Solve
Track No.
1 55
2 43
3 21
4 176
5 189
6 123
Output Sequence :
[50, 43, 21, 0, 199, 189, 176, 123, 55]
The total seek time is : 393 ms.
Average Seek Time : 49.125 ms.
Generate Graph
Seek Time Comparison
Compare Algo
Explaination
1 Initially R/W head is at 50 .
2 As algorithm is C-SCAN it will go from 50 to 43 .
3 As algorithm is C-SCAN it will go from 43 to 21 .
4 As algorithm is C-SCAN it will go from 21 to 0 .
5 As algorithm is C-SCAN it will go from 0 to 199 .
6 As algorithm is C-SCAN it will go from 199 to 189 .
7 As algorithm is C-SCAN it will go from 189 to 176 .
8 As algorithm is C-SCAN it will go from 176 to 123 .
9 As algorithm is C-SCAN it will go from 123 to 55 .

Figure 1


C-SCAN
Track Numbers
Order of Arrival
0
25
50
75
100
125
150
175
200
0
1
2
3
4
5
6
7
8

Example 2


Disk Scheduling Algorithms
C-SCAN
Change Theme
About Algorithm
Enter Number of Tracks :
5
GO
Intial R/W Head Position :
50
Range : 0 - 199
Enter Direction :
right
Reset
Solve
Track No.
1 134
2 179
3 89
4 21
5 16
Output Sequence :
[50, 89, 134, 179, 199, 0, 16, 21]
The total seek time is : 369 ms.
Average Seek Time : 52.714285714285715 ms.
Generate Graph
Seek Time Comparison
Compare Algo
Explaination
1 Initially R/W head is at 50 .
2 As algorithm is C-SCAN it will go from 50 to 89 .
3 As algorithm is C-SCAN it will go from 89 to 134 .
4 As algorithm is C-SCAN it will go from 134 to 179 .
5 As algorithm is C-SCAN it will go from 179 to 199 .
6 As algorithm is C-SCAN it will go from 199 to 0 .
7 As algorithm is C-SCAN it will go from 0 to 16 .
8 As algorithm is C-SCAN it will go from 16 to 21 .

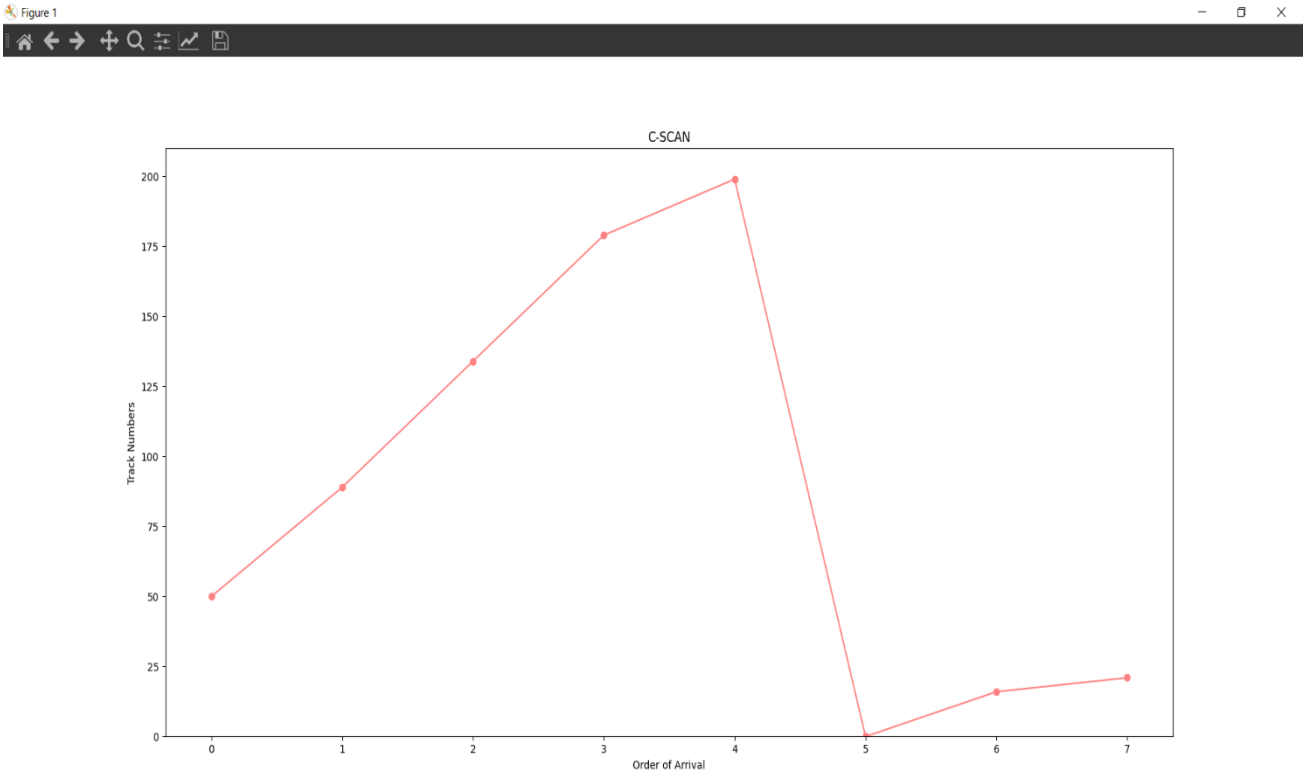


Figure 1
C-SCAN
Track Numbers
Order of Arrival
0
25
50
75
100
125
150
175
200
0
1
2
3
4
5
6
7

LOOK

Introduction

It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus, it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Advantages

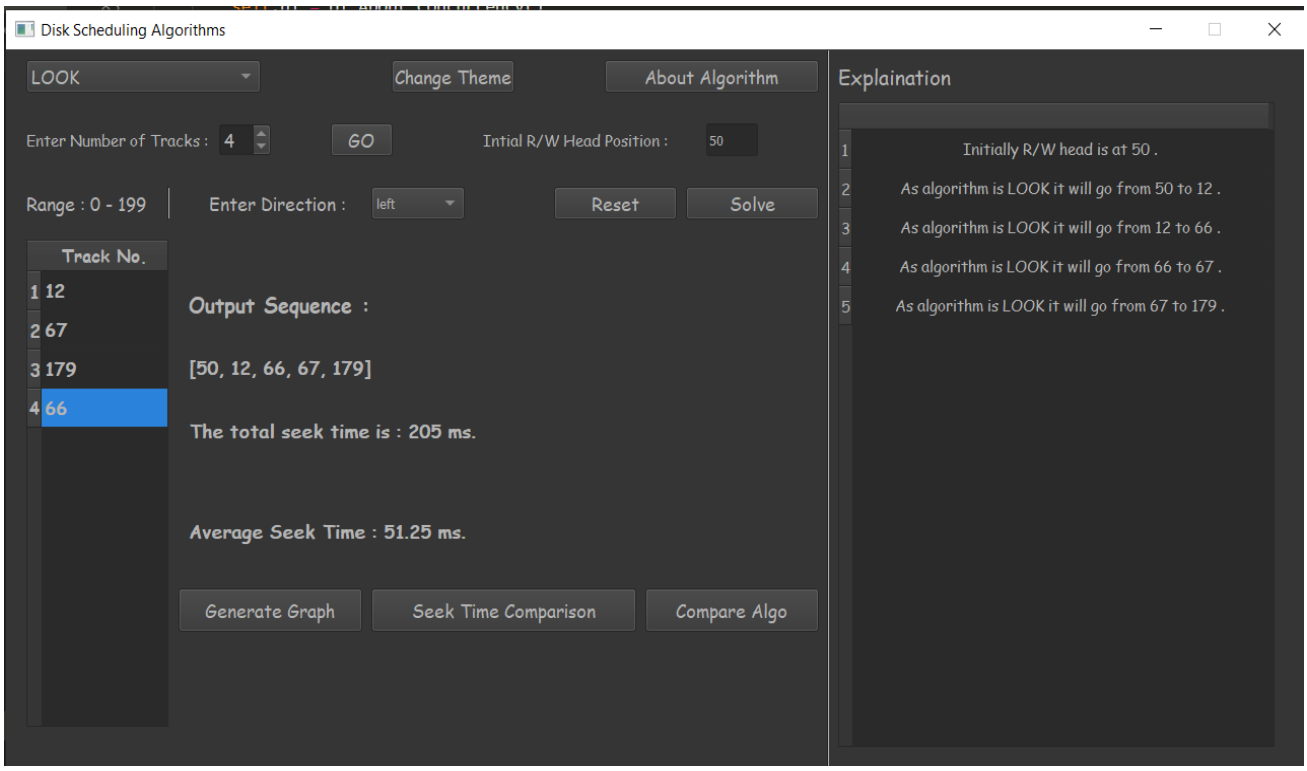
- If there are no requests left to be services the Head will not move to the end of the disk unlike SCAN algorithm.
- Better performance is provided compared to SCAN Algorithm.
- Starvation is avoided in LOOK scheduling algorithm.
- Low variance is provided in waiting time and response time.

Disadvantages

- Overhead of finding the end requests is present.
- Cylinders which are just visited by Head have to wait for long time.

Snapshots

Example 1



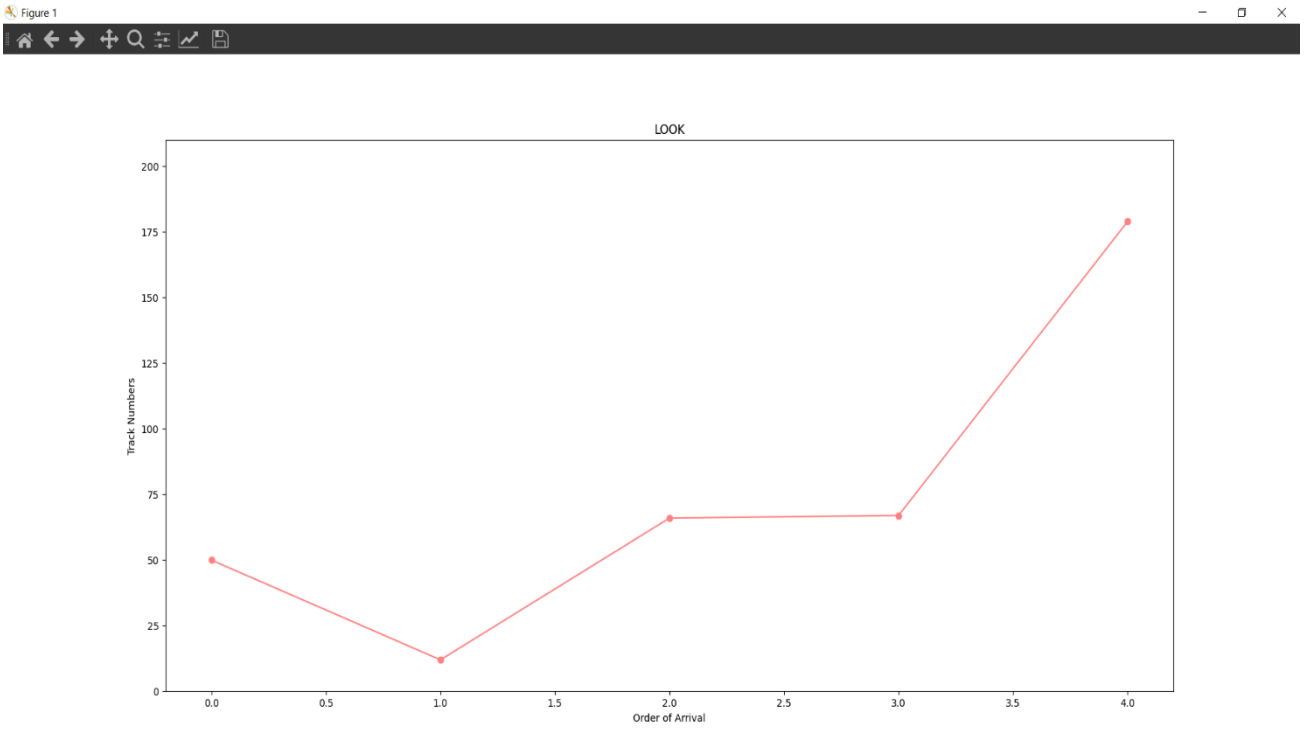
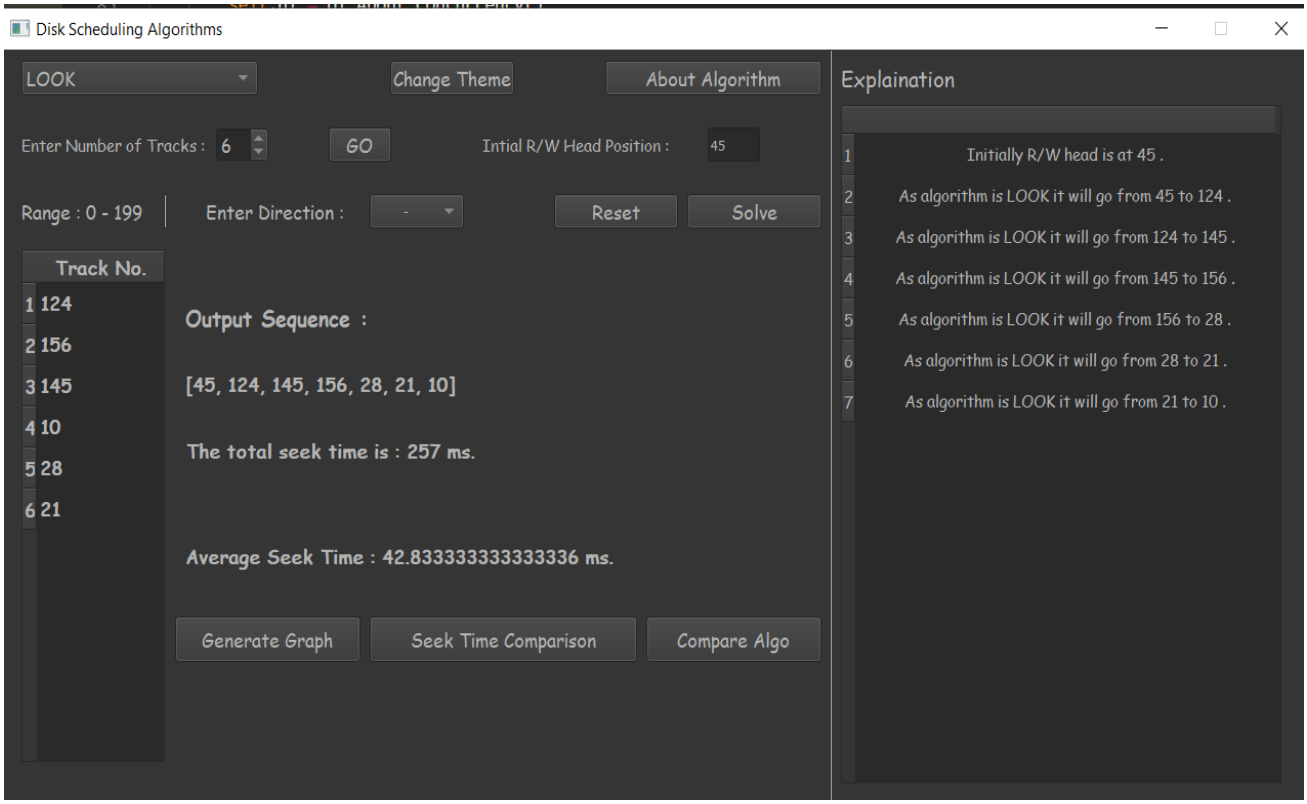
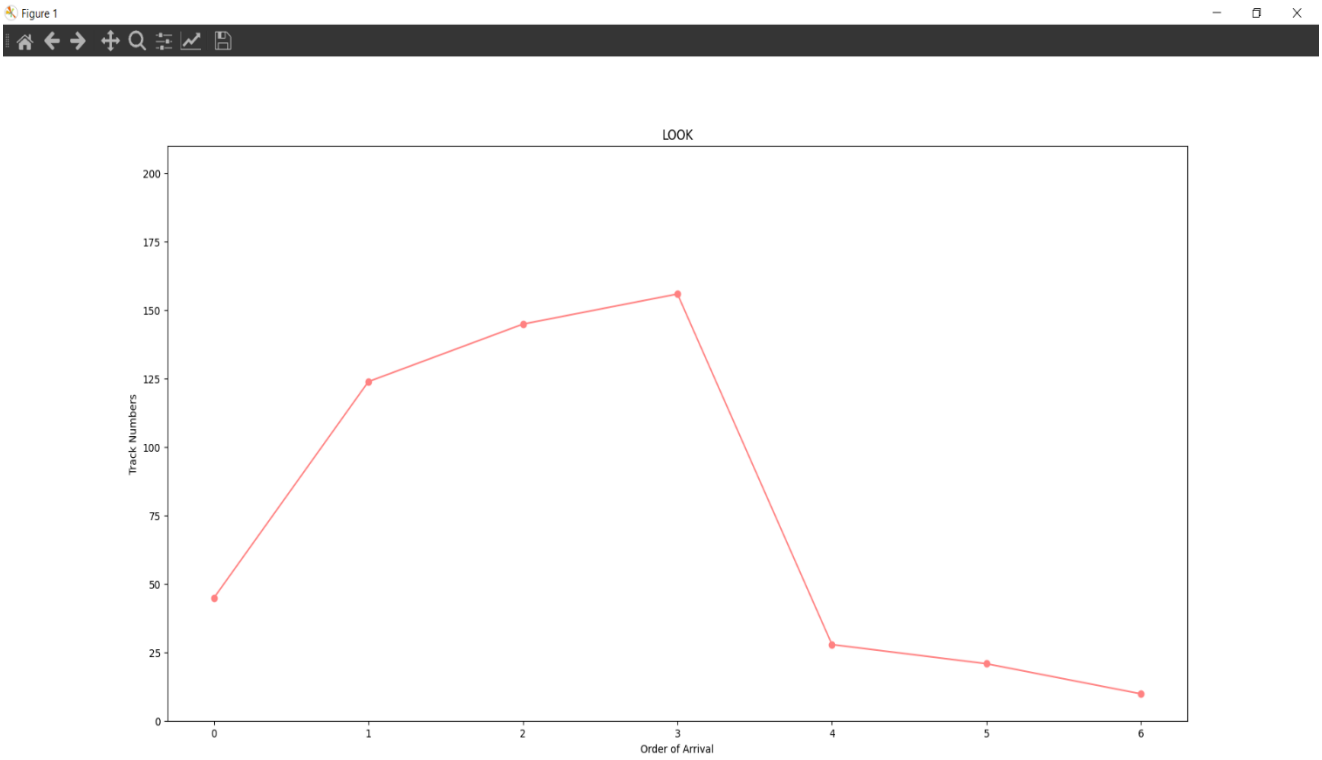


Figure 1
LOOK
Track Numbers
Order of Arrival
0
25
50
75
100
125
150
175
200
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Example 2


Disk Scheduling Algorithms
LOOK
Change Theme
About Algorithm
Explaination
Enter Number of Tracks : 6
GO
Intial R/W Head Position : 45
Range : 0 - 199
Enter Direction :
Reset
Solve
Track No.
1 124
2 156
3 145
4 10
5 28
6 21
Output Sequence :
[45, 124, 145, 156, 28, 21, 10]
The total seek time is : 257 ms.
Average Seek Time : 42.833333333333336 ms.
Generate Graph
Seek Time Comparison
Compare Algo
1 Initially R/W head is at 45 .
2 As algorithm is LOOK it will go from 45 to 124 .
3 As algorithm is LOOK it will go from 124 to 145 .
4 As algorithm is LOOK it will go from 145 to 156 .
5 As algorithm is LOOK it will go from 156 to 28 .
6 As algorithm is LOOK it will go from 28 to 21 .
7 As algorithm is LOOK it will go from 21 to 10 .



C-LOOK

Introduction

As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Advantages

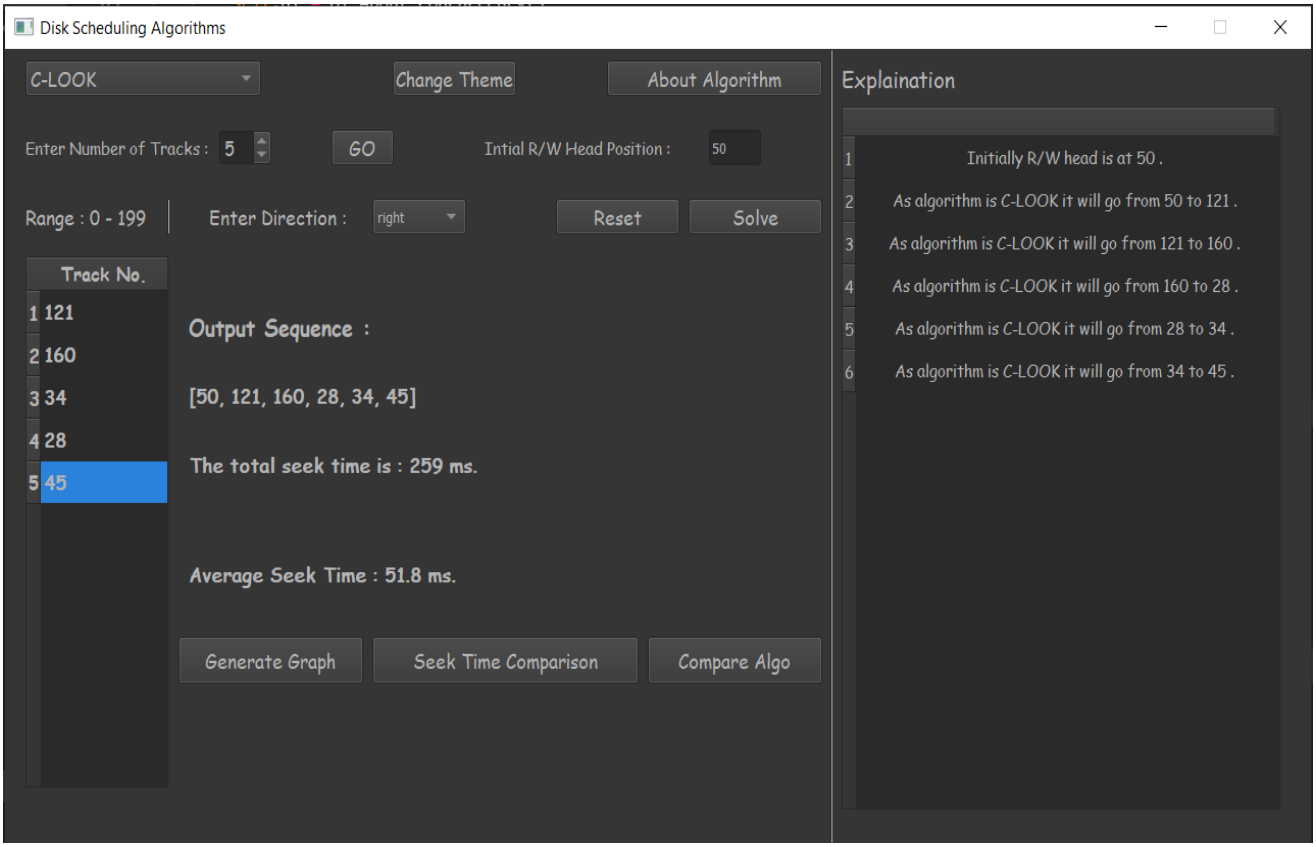
1. In C-LOOK the head does not have to move till the end of the disk if there are no requests to be serviced.
2. There is less waiting time for the cylinders which are just visited by the head in C-LOOK.
3. C-LOOK provides better performance when compared to LOOK Algorithm.
4. Starvation is avoided in C-LOOK.
5. Low variance is provided in waiting time and response time.

Disadvantages

- In C-LOOK an overhead of finding the end requests is present.

Snapshots

Example 1


Disk Scheduling Algorithms
C-LOOK
Change Theme
About Algorithm
Explaination
Enter Number of Tracks :
5
GO
Intial R/W Head Position :
50
Range : 0 - 199
Enter Direction :
right
Reset
Solve
Track No.
1 121
2 160
3 34
4 28
5 45
Output Sequence :
[50, 121, 160, 28, 34, 45]
The total seek time is : 259 ms.
Average Seek Time : 51.8 ms.
Generate Graph
Seek Time Comparison
Compare Algo
1 Initially R/W head is at 50 .
2 As algorithm is C-LOOK it will go from 50 to 121 .
3 As algorithm is C-LOOK it will go from 121 to 160 .
4 As algorithm is C-LOOK it will go from 160 to 28 .
5 As algorithm is C-LOOK it will go from 28 to 34 .
6 As algorithm is C-LOOK it will go from 34 to 45 .

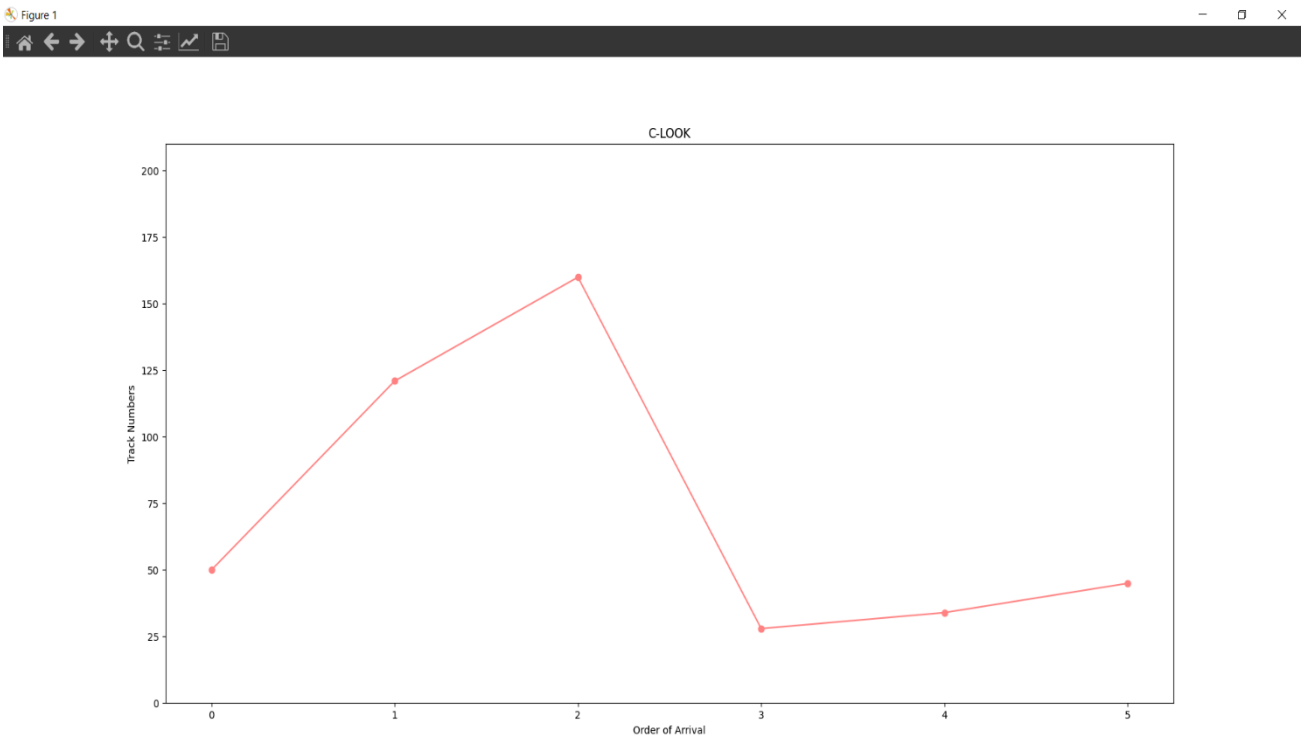
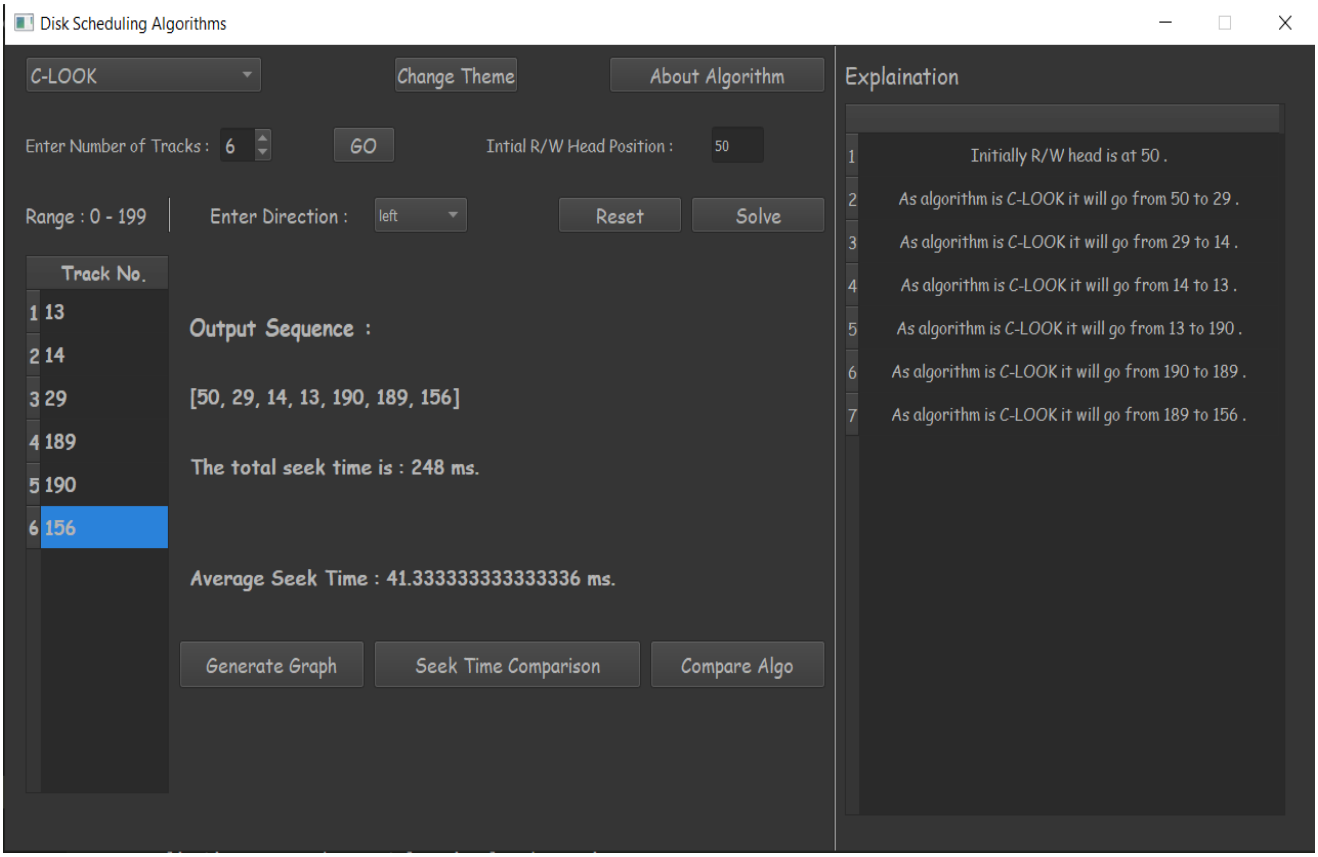


Figure 1
C-LOOK
Track Numbers
Order of Arrival
0
25
50
75
100
125
150
175
200
0
1
2
3
4
5

Example 2


Disk Scheduling Algorithms
C-LOOK
Change Theme
About Algorithm
Enter Number of Tracks : 6
GO
Intial R/W Head Position : 50
Range : 0 - 199
Enter Direction : left
Reset
Solve
Track No.
1 13
2 14
3 29
4 189
5 190
6 156
Output Sequence :
[50, 29, 14, 13, 190, 189, 156]
The total seek time is : 248 ms.
Average Seek Time : 41.333333333333336 ms.
Generate Graph
Seek Time Comparison
Compare Algo
Explaination
1 Initially R/W head is at 50 .
2 As algorithm is C-LOOK it will go from 50 to 29 .
3 As algorithm is C-LOOK it will go from 29 to 14 .
4 As algorithm is C-LOOK it will go from 14 to 13 .
5 As algorithm is C-LOOK it will go from 13 to 190 .
6 As algorithm is C-LOOK it will go from 190 to 189 .
7 As algorithm is C-LOOK it will go from 189 to 156 .

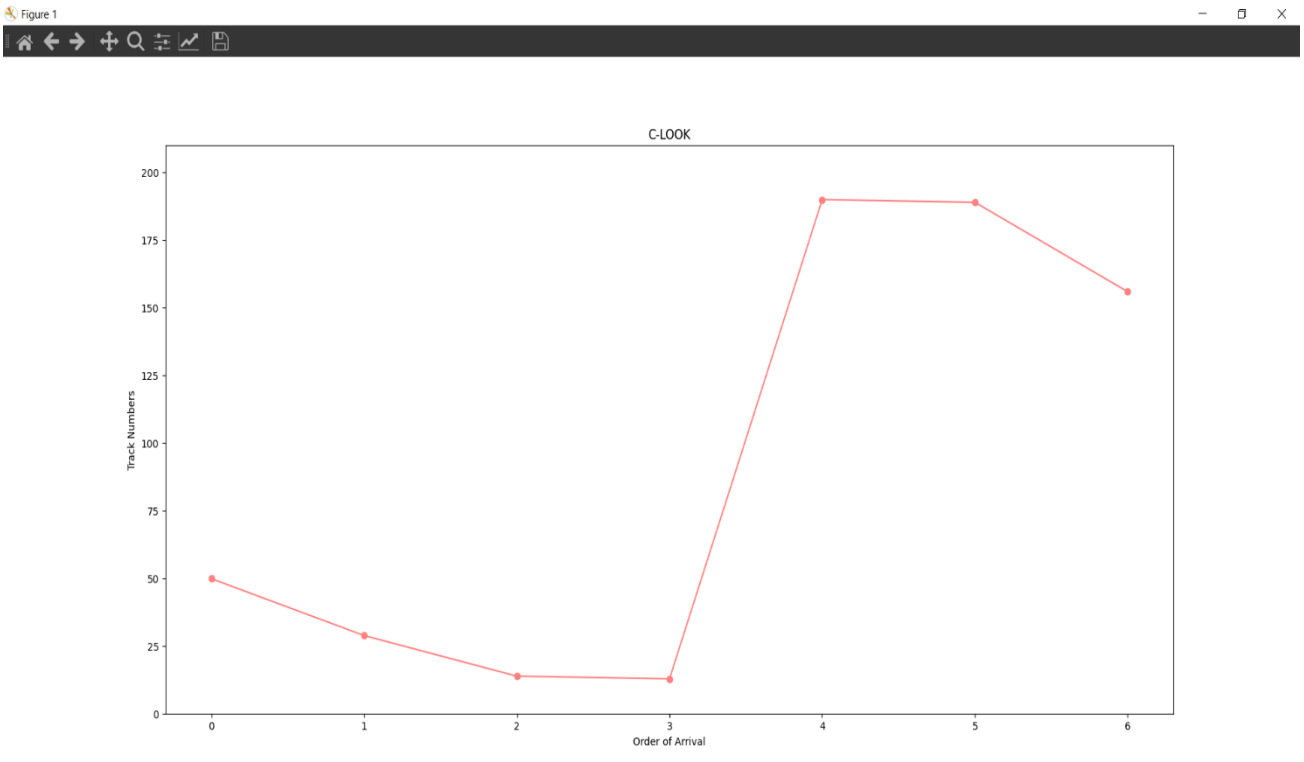


Figure 1
C-LOOK
Track Numbers
Order of Arrival
0
25
50
75
100
125
150
175
200
0
1
2
3
4
5
6

LIFO

Introduction

In LIFO (Last In, First Out) algorithm, newest jobs are serviced before the existing ones i.e., in order of requests that get serviced the job that is newest or last entered is serviced first and then the rest in the same order.

Advantages

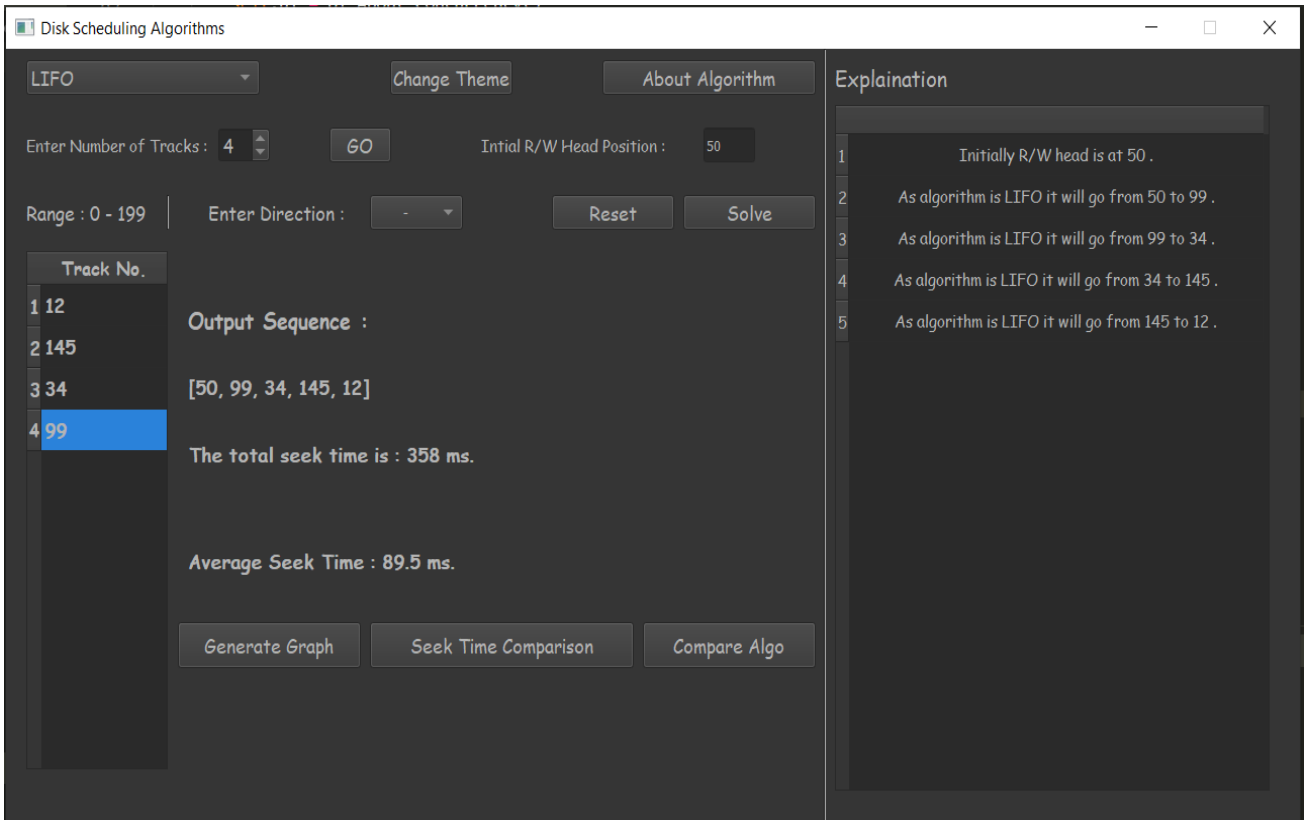
- Maximizes locality and resource utilization

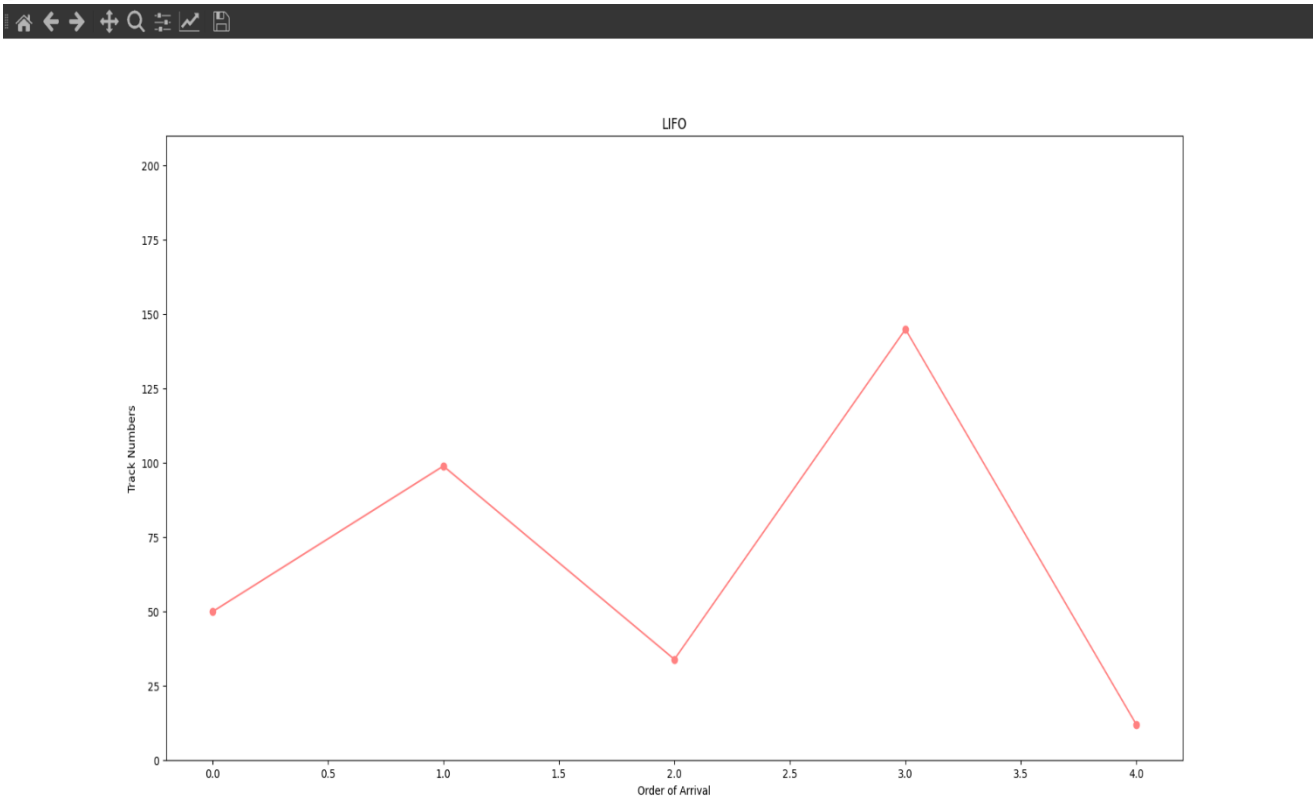
Disadvantages

- Can seem a little unfair to other requests and if new requests keep coming in, it cause starvation to the old and existing ones.

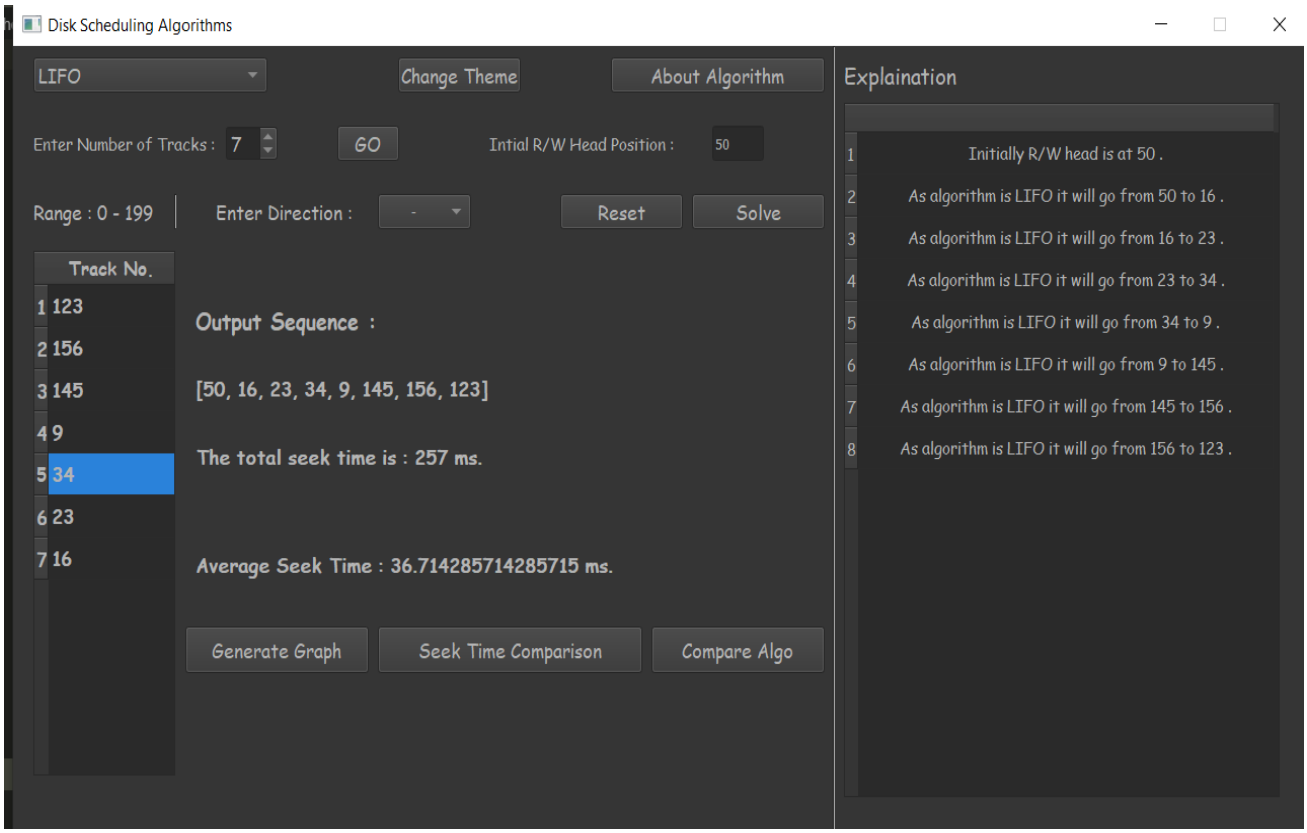
Snapshots

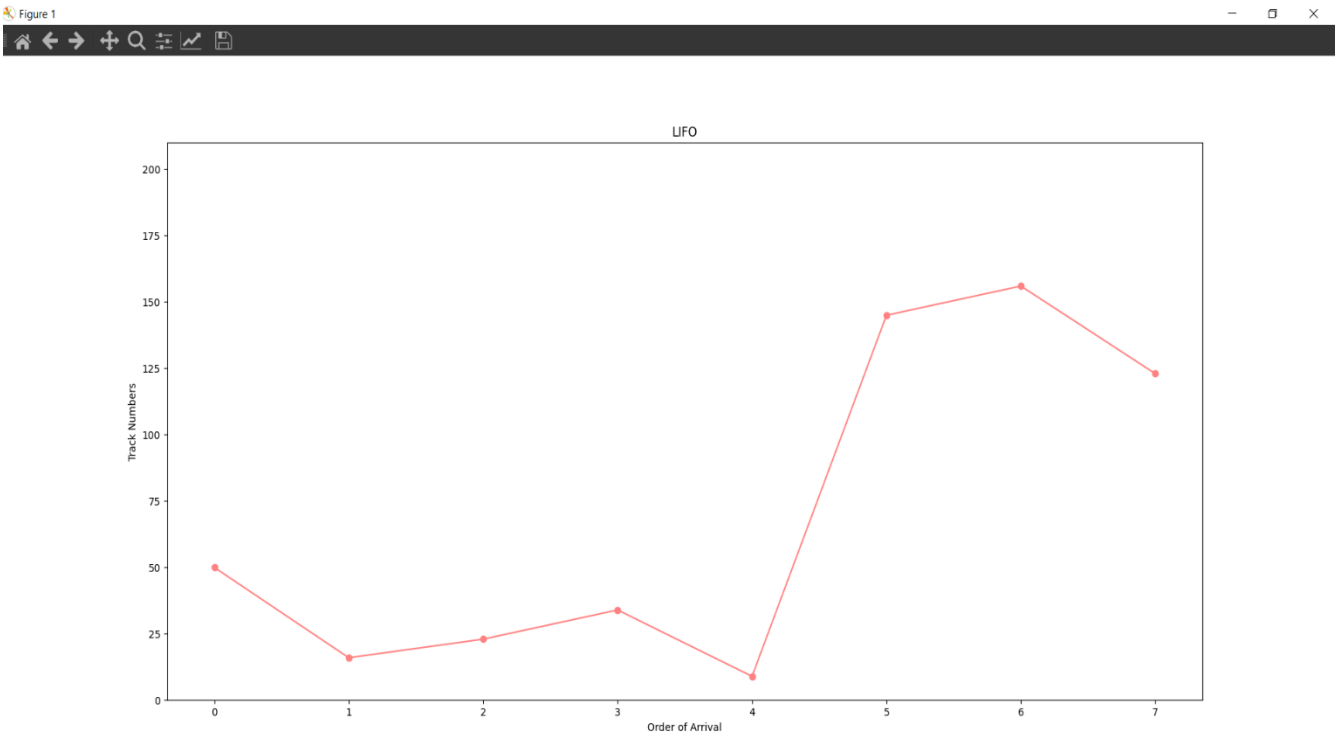
Example 1




LIFO
Track Numbers
Order of Arrival
200
175
150
125
100
75
50
25
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Example 2


Disk Scheduling Algorithms
LIFO
Change Theme
About Algorithm
Explaination
Enter Number of Tracks : 7
GO
Intial R/W Head Position : 50
Range : 0 - 199
Enter Direction :
Reset
Solve
Track No.
1 123
2 156
3 145
4 9
5 34
6 23
7 16
Output Sequence :
[50, 16, 23, 34, 9, 145, 156, 123]
The total seek time is : 257 ms.
Average Seek Time : 36.714285714285715 ms.
Generate Graph
Seek Time Comparison
Compare Algo
1 Initially R/W head is at 50 .
2 As algorithm is LIFO it will go from 50 to 16 .
3 As algorithm is LIFO it will go from 16 to 23 .
4 As algorithm is LIFO it will go from 23 to 34 .
5 As algorithm is LIFO it will go from 34 to 9 .
6 As algorithm is LIFO it will go from 9 to 145 .
7 As algorithm is LIFO it will go from 145 to 156 .
8 As algorithm is LIFO it will go from 156 to 123 .

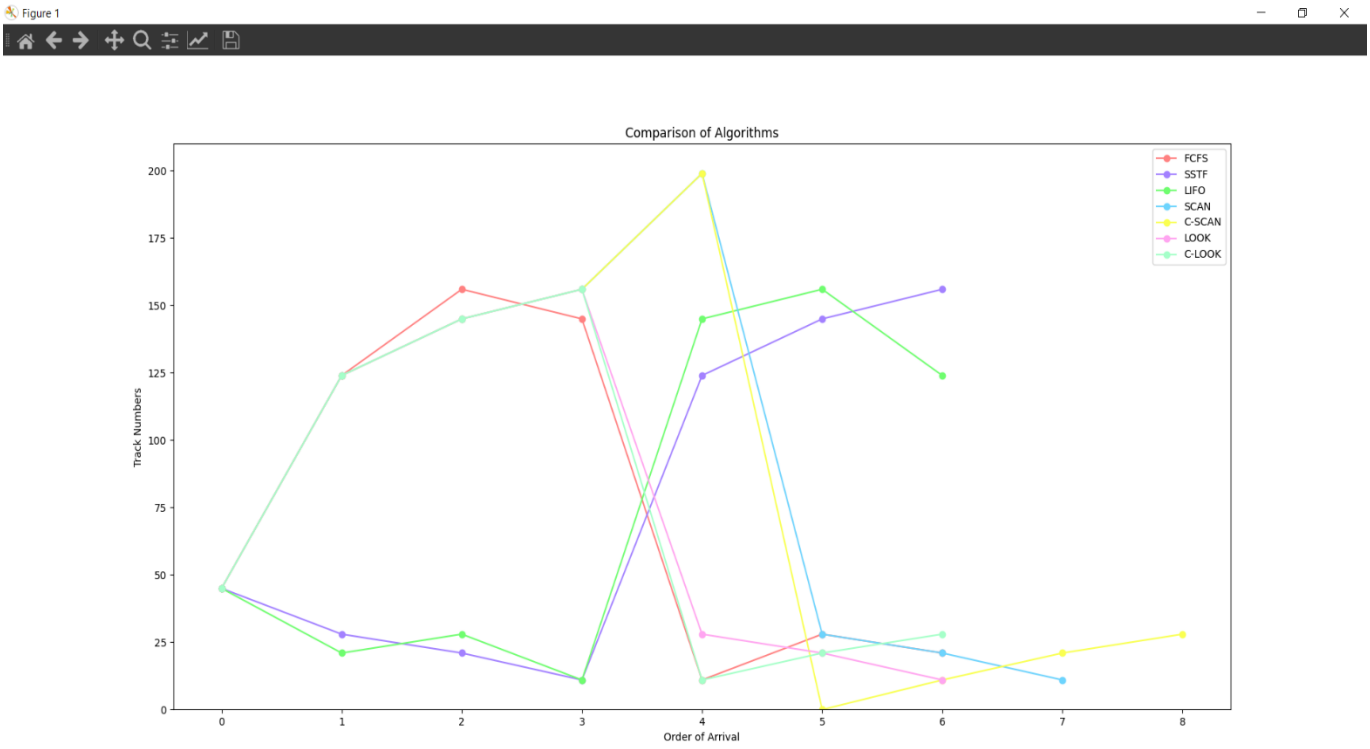


Comparison of All Algorithms

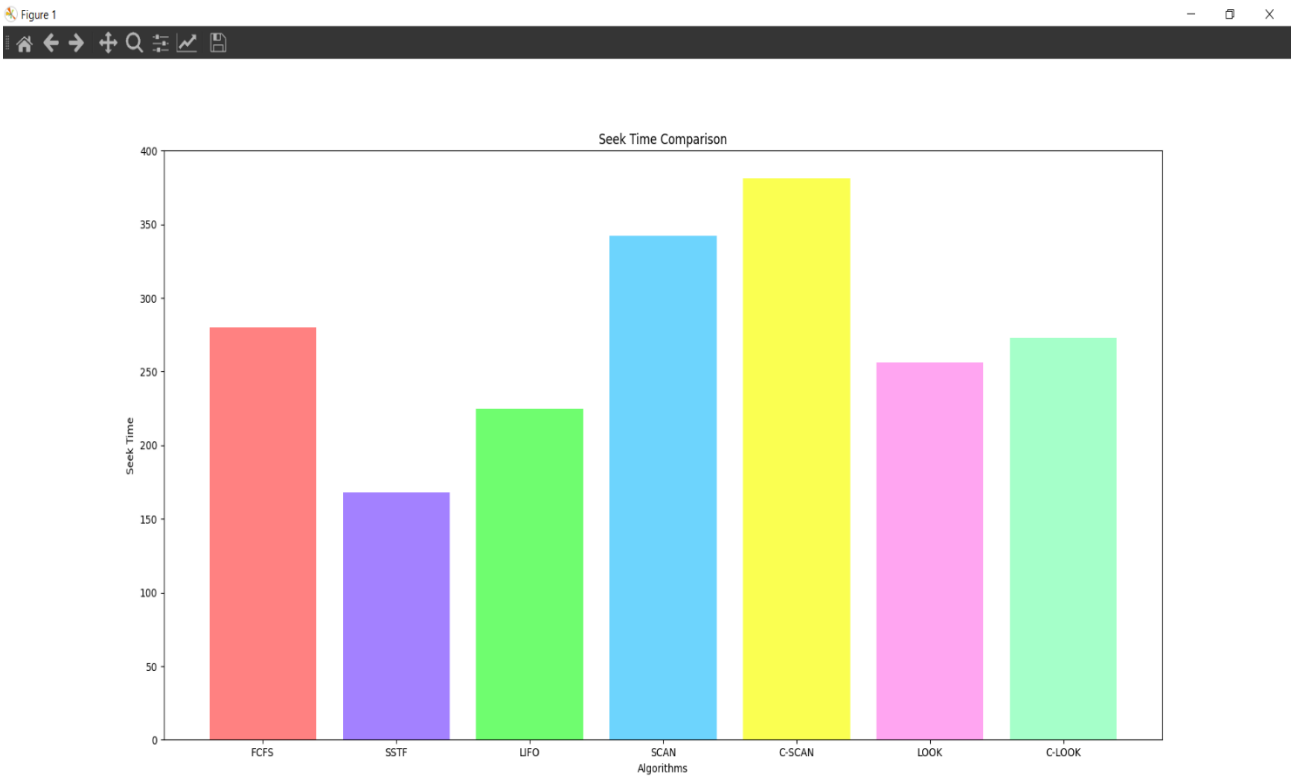
Example 1

Input String: 124 156 145 11 28 21

1. Compare all Algorithms Graph



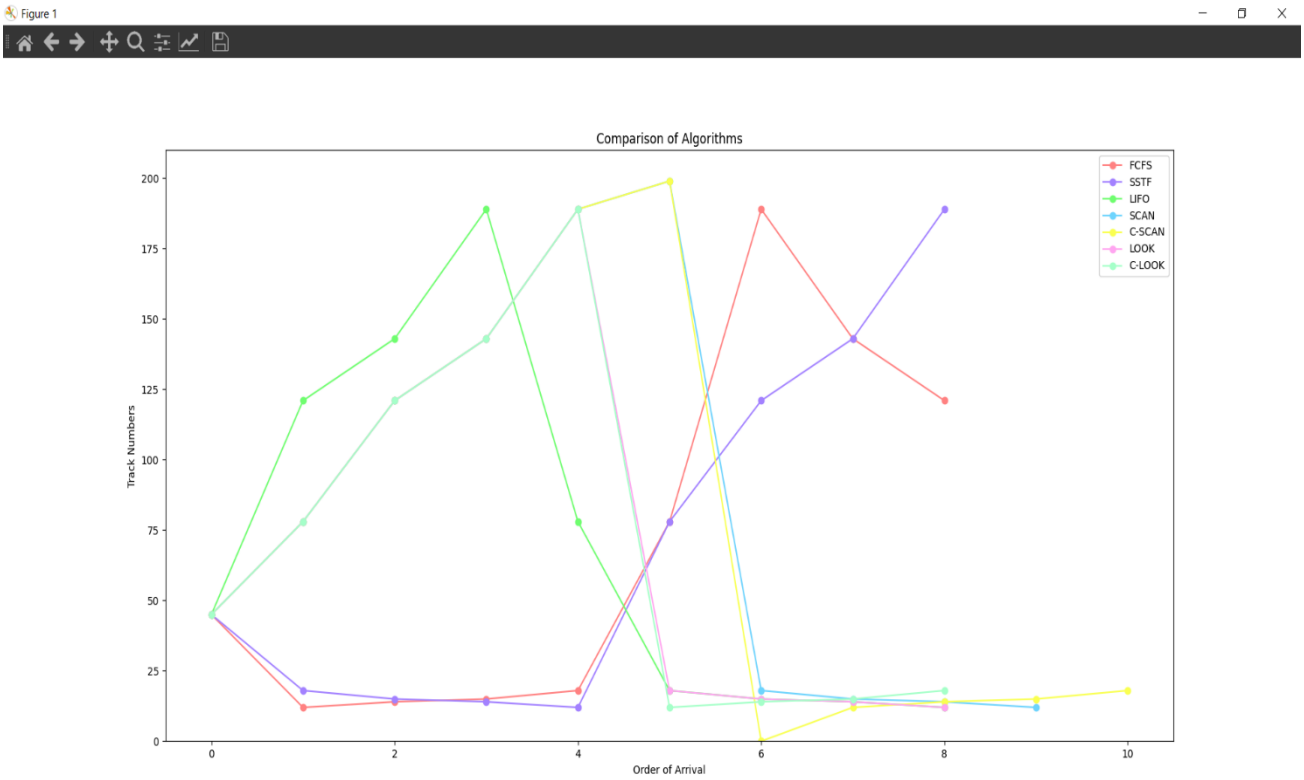
2. Seek Time Comparison of all Algorithms Graph



Example 2

Input String: 12 14 15 18 78 189 143 121

1. Compare all Algorithms Graph



2. Seek Time Comparison of all Algorithms Graph

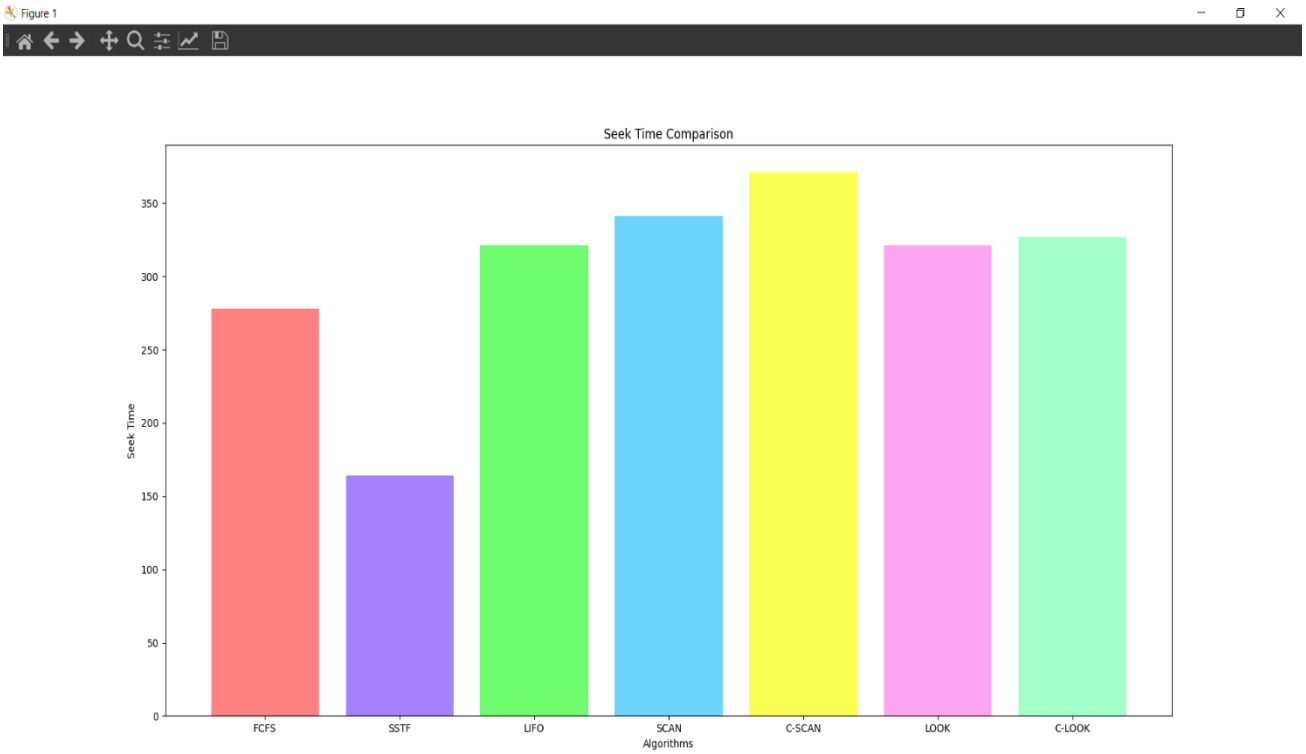


Figure 1
Seek Time Comparison
Seek Time
0
50
100
150
200
250
300
350
FCFS
SSTF
LIFO
SCAN
C-SCAN
LOOK
C-LOOK
Algorithms